FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Agile Methodologies in Mission Critical Software Development and Maintenance

Antonieta Ponce de Leão

Integrated Master in Informatics and Computing Engineering

Supervisor: João Carlos Pascoal Faria, Ph.D.

June, 2014

© Antonieta Ponce de Leão, 2014

Agile Methodologies in Mission Critical Software Development and Maintenance

Maria Antonieta Dias Ponce de Leão e Oliveira

Integrated Master in Informatics and Computing Engineering

Approved in public trials, by jury:

President: Nuno Honório Rodrigues Flores (Ph.D.)

External Vowel: Alberto Manuel Rodrigues da Silva (Ph.D.)

Supervisor: João Carlos Pascoal Faria (Ph.D.)

July, 17th 2014

Abstract

Software development depends on people and because of this many issues arise specially regarding consistency and quality. Guaranteeing quality, accurate estimation, and high-performance knowledge workers is not an easy job, and for this there are many methodologies and processes that help.

This is not a new issue; many software development companies, somewhere along their lifetime passed through this hassle, and with the study of their victories and defeats as well as worldwide recognized best practices, were able to improve their processes and performance.

ALERT, is an example of a software development company that is continuously striving to improve, and their goal, at the moment, is to increase quality, productivity and estimation accuracy.

The goal of this project is to identify any exiting issues in the current software development processes and practices at the organization, propose a hybrid and customized proposal of agile and classical methodologies that will help to optimize the performance of the knowledge workers, never forgetting that the productivity of knowledge workers is deeply connected with their motivation.

This project was developed in three main phases, diagnosis, proposal and implementation of the proposal in three pilot teams.

With the information gathered from individual interviews and close observation, a diagnosis was created that represented the company status at that moment.

Based on the diagnosis, the company's past experiences, and well known models, methodologies and processes the scope was defined - Scrum, Kanban, TSP and CMMI - and a proposal was created.

The goals of the proposal were created in order to hopefully increase productivity, reduce defects and/or improve estimation accuracy, expediting the software development process.

In the end it is presented an analysis of the subjective and objective results of the proposal's impact in the pilot teams. Nevertheless the main outputs of this dissertation are the diagnosis and the customized proposal, as the pilot's project success is highly dependent on the participating knowledge workers.

Resumo

O desenvolvimento de *software* depende essencialmente das pessoas e isso faz com que surjam diversas questões, especialmente em relação à qualidade e consistência. Garantir a qualidade, estimativas precisas, e profissionais intelectuais de alto desempenho não é uma tarefa fácil, e para isso existem muitas metodologias e processos que ajudam.

Este não é um assunto novo pois muitas empresas de desenvolvimento de *software*, em algum momento ao longo de sua vida, passaram por esse incómodo, e com o estudo das suas vitórias e derrotas, e com a aplicação das melhores práticas reconhecidas a nível mundial, foram capazes de melhorar os seus processos e o seu desempenho.

A ALERT é um exemplo de uma empresa de desenvolvimento de *software* que se esforça continuamente para melhorar, e o seu objetivo, de momento, é aumentar a qualidade, a produtividade e a precisão das estimativas.

O objetivo deste projeto é identificar quaisquer problemas existentes nos processos e práticas de desenvolvimento de *software* atuais da organização, propor uma proposta híbrida e personalizada de metodologias ágeis e clássicas que irão ajudar a otimizar o desempenho dos trabalhadores, nunca esquecendo que a produtividade de profissionais intelectuais está correlacionada com a sua motivação.

Este projeto foi realizado em três grandes fases, diagnóstico, proposta e implementação da proposta em três equipas piloto.

Através da informação recolhida nas entrevistas individuais e observação atenta, foi criado um diagnóstico que representa o estado da empresa naquele momento.

Com base no diagnóstico, nas experiências passadas da empresa, e em modelos bem conhecidos, metodologias e processos, foi definido um âmbito – Scrum, Kanban, TSP e CMMI – e a proposta elaborada.

Os objetivos da proposta foram criados na expectativa de aumentar a produtividade, reduzir defeitos e/ou melhorar a precisão das estimativas, agilizando o processo de desenvolvimento de software.

No final, é apresentada uma análise dos resultados subjetivos e objetivos resultantes do impacto da proposta nas equipas-piloto. No entanto, os principais *outputs* desta dissertação são o diagnóstico e a proposta personalizada, uma vez que o sucesso do projeto piloto depende essencialmente dos seus participantes.

Acknowledgements

First of all I would like to thank my supervisor for all the support and guidance throughout the whole dissertation and preparation for the dissertation.

I would also like to thank my superior, Rui Borges, first for creating the opportunity for this dissertation to happen and for trusting my judgment and believing in me.

Then I would like to thank to every knowledge worker that found the time and the will to participate in my interviews, for their honesty and openness to share all their concerns, opinions, doubts and wishes, without their participation this project would never started.

I would like to thank particularly to all the knowledge workers that participated in the pilot program and that trusted me to guide them, and that made me feel wise.

To my mother and father, for always inspiring me in research and to be always hungry for knowledge.

To my teacher of Personal and Interpersonal Proficiency, Manuel Firmino Ph.D., for all the wisdom that he though us in his classes.

To my teacher of Soft Skills: Leadership and Team Management, Cristina Nunes de Azevedo, Ph.D. for all the knowledge on teams and leadership.

To my dear great friends that were always there for me in the sunny days and in the hurricane days, a deep thank you to them. I hope we grow old like this.

And at last, but not least, I want to thank my dear boyfriend you put up with my tiredness, bad mood, long hours, and weekends and holidays extra work, without you our home would have fallen. Thank you for taking care of our little family Sushi and Chocolate, which were always so excited to see me coming home, and that would be always with me as my desktop background, so every time I was tired or frustrated I would look to their pictures and smile.

Thank you all, this project would never have been what it is without you.

<Antonieta Ponce de Leão>

In accordance with the terms of the master's thesis contract in business environment and the confidentiality agreement executed with ALERT Life Sciences Computing, S.A. ("ALERT"), this report is confidential and may contain references to inventions, know-how, drawings, computer software, trade secrets, products, formulas, methods, plans, specifications, projects, data or works protected by ALERT's industrial and/or intellectual property rights. This report may be used solely for research and educational purposes. Any other kind of use requires prior written consent from ALERT.

Contents

List of Figures

List of Tables

Abbreviations

BU	Business Unit
CM	Configuration Management
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
COQ	Cost-of-Quality
JIT	Just-In-Time
MA	Measurement and Analysis
OCM	Organizational Change Management
OOPSLA	Object-Oriented Programming, Systems, Languages & Applications
PMC	Project Monitoring and Control
PP	Project Planning
PPQA	Process and Product Quality Assurance
PROBE	PROxy Based Estimation
PSP	Personal Software Process
QC	Quality Control
REQM	Requirements Management
SAM	Supplier Agreement Management
SEI	Software Engineering Institute
SPM	Senior Product Manager
TSP	Team Software Process
UP	Unified Process
WBS	Work Breakdown Structure
WIP	Work In Progress
XP	Xtreme Programing

Chapter 1

Introduction

Software development was always challenged, because “from the very beginning of the computer era, software products have been delivered much later than planned, have cost much more than expected, and have been highly defective” (Humphrey and Over 2010).

The machine behind software development is the people, the developers, testers, analysts, software architects, team leaders and managers, they are the wheels that make the machine turn, without them there is not software development.

In many aspects we can say that software development is handmade (Tonini, de Carvalho, and de Mesquita Spinola 2008), because every developer has his quirks and twerks making everything he creates unique and different; this makes software development extremely hard to manage.

The fact that software development depends on people, the fact that software projects are troubled and that knowledge work management is in many ways peculiar, allied with the fact that software development processes are essential for an organization to obtain the required levels of productivity and quality (Carvalho et al. 2012) represent some of the points why this dissertation is so important.

1.1 Context

Due to the many issues and difficulties of knowledge work management many processes where created to try to make the software development process more manageable. We can go from the classical to the agile, passing through Waterfall to RUP to Xtreme Programing, Scrum and so on, without ever finding a solution that works. This is why, in many cases, we have to adapt the process to the problem. Some projects will work much better in Scrum as opposed to Waterfall and in others Waterfall may bring the discipline necessary to finish the project.

So based on the issues of software development, this thesis aims to help in the finding of an optimized solution to software teams inside a software company that develops and maintains mission critical software.

1.2 Motivation and Objectives

To describe the main objective of this thesis in one sentence would be: Help the organization's software development teams to improve their performance, through the adoption of good practices based on agile and classical methodologies.

The success and failure of software problems and the challenge related to knowledge workers management and motivation is not a new issue, but is still unsolved. The reason why is because there are no magical recipes that can be applied, just like in the book, and solve all of these problems.

Companies and knowledge workers are deeply affected by their surroundings, their culture and their DNA. As so to successfully implement new methodologies, processes or methods it is necessary to understand the surroundings, the culture, the DNA.

In an initial stage, previous to the interviews, some issues like usual employee turnover, the constant accumulation of technical debt[1], and the lack of team ownership of the projects predefined the initial scope: CMMI level 2, Scrum, and Personal Software Process/Team Software Process (PSP/TSP). However during the interviews stage it was clear that the inclusion of Kanban in the proposal would make much sense.

CMMI level 2 is a company's objective, Scrum was chosen because it seems to fit the large necessity that the company has regarding flexibility and TSP because one of the main focus of the objective of this dissertation is the product's quality. Kanban is suited for managing continuous maintenance work, and the company's main product is, mainly, in the maintenance phase. With this four referential a proposal was created and to accelerate the feedback "response time" of the proposal frequent lessons learned sessions where conducted.

The main challenge of this thesis is that is totally dependent on individuals, and to motivate the teams and managers involved in the process, and to always have the sponsorship of high and middle management will be challenging.

People are the main challenge because they will need constant motivation; they will need support to implement the proposal and to never give up. And specially help to overcome their skepticism regarding the success of the whole project.

It was also challenging to incorporate the various referentials, to find a common ground between Scrum, Kanban, and TSP at a micro level and guarantying it would not affect, and even help the CMMI level 2 implementation that acts in a higher level.

[1] Eventual consequences of poor software – short term solutions, poor architecture etc

Another big challenge of this dissertation is the impact evaluation. Gathering and analyzing subjective metrics will be exhausting and analyzing success in objective metrics might not be possible due to the short time frame of the project.

The main outputs that arise from this thesis are: the diagnosis, understand what the teams do good and not so good; the proposal, that will be customized and focused on the problems that the teams face; and the impact evaluation of the proposal that may be difficult to measure, because, as was said before, this project depends deeply on the people involved in it and on my leadership capacity of helping them through the process.

1.3 Project

The idea of this project is not to blindly adopt a new methodology to increase knowledge workers performance. As so this project was divided in three main phases, which lasted, in total, 20 weeks. The three main phases where the diagnosis, the proposal and the pilot implementation.

To perform an accurate diagnosis, in the first 3 weeks I conducted a series of interviews to more than 70 of the organization's knowledge workers. The interviews were personal and completely confidential so that everyone felt comfortable enough to be honest, to say "The Good, the Bad and the Ugly" of software development inside the organization.

To complement the data gathered form the interviews I also observed knowledge workers in their "habitat" and tried to uncover other underlying issues, because, as can be imagined, successfully gather accurate and precise data and testimony from people that do not know you, and that consider you an outsider may be extremely difficult.

After compiling and analyzing all of this information I elaborated a diagnosis of the organization's status, at that moment, regarding the software development process and methodologies.

Based on the diagnosis, and through extensive research I created a proposal, which was customized to the diagnosis. As this project had a limited time frame, the proposal could not address all the problems reported in the diagnosis. As the company evolves, this proposal will also evolve, some choices that would not be accepted at that moment, will certainly be in the feature.

As the proposal was approved we implemented it in 3 pilot teams, through the course of nearly 10 weeks. In the end, all the participating knowledge workers were interviewed a second time to gather inside data from the impact of the complete proposal, to measure their satisfaction and to evaluate the level of success of the proposal.

"To mature software engineering, we must learn from history, and to learn from history, we must define what we do when we develop software and how we do it." (Humphrey 2008)

1.4 Critical Success Factors

All the success factors round up to people, the success of this project will depend only on the people, the ones participating in it, the ones sponsoring it and the "outsiders" in some ways connected with the core group.

This is a very challenging project because it involves the commitment of all the intervenient and their interest that this project becomes a success.

Changing the way people do their job it's not an easy task, we have to fight the skepticism of the knowledge workers, which already tried and failed in previous process reformulations. Fight with the unwilliness that will certainly come from members, and strive to always have the management sponsorship even when tough decisions take place.

"Introducing self-directed teamwork and rational management requires a major cultural change and it can only be accomplished with active senior-management leadership and support." (Humphrey 2007)

The teams will need a strong leader to keep them on track, and support them in the issues that will certainly arise. They will need motivation to never give up, and the thing about motivation is that it's not black and white. People may feel motivated from very different things. In creative work money only works to a certain point. If you pay people enough that they do not need to think about money, if they are comfortable with their salary, money gets off the table and people can focus on their work. (Pink 2011)

For the success of this project it is also very important that an environment of open and creative communication is created, so that good ideas can flow, team members can give feedback without fear of reprisals and bad ideas can be quickly dismissed.

It is important that the higher management is also involved in the project, so teams and me feel their sponsorship and feel comfortable enough to take the necessary risks to achieve success.

This topic is particularly important, since at the beginning much time will be spent training the teams and learning the process, and if teams don't feel sponsorship they'll fail to try to apply the process because they will be afraid of wasting time. So it is very important that higher management supports this project and has the vision to see the big picture involved in it. A few hours spent at the beginning will mean a bunch of hours not wasted in the long run.

"Jerry Weinberg puts it nicely: "Things are the way they are because they got that way". Or here's another one (don't know who said it first): "If all you ever do is all you've ever done, then all you'll ever get is all you ever got"."(Kniberg 2011)

1.5 Report Structure

This report is composed of 7 chapters, being the first one the Introduction where the whole project is briefly detailed.

The state-of-art review describes Scrum, and TSP, focusing on their possible combination with CMMI level 2. Kanban was also included and researched as an important part of this dissertation.

In the third chapter it is described the diagnosis conducted to the company, focusing points like: communication; work methodology; evaluation; planning; and quality.

The proposal described in the fourth chapter is customized to this diagnosis, and focused essentially on Scrum and Kanban, taking into account good practices from TSP, never forgetting the goal of CMMI level 2.

In the fifth chapter it is described the pilot implementation detailing the teams, all the encountered difficulties and the lessons learned. This chapter is followed by an analysis of the results, were it is briefly detailed the overcome challenges, and the subjective and objective results gathered throughout the pilot implementation.

Finally in chapter seven the conclusions and objectives satisfaction are detailed, also mentioning the future work and known challenges.

Chapter 2

Related work

This chapter represents the state-of-art review to study the field that this thesis belongs to. It will provide insight to the lessons learned from other studies and projects and will help to achieve a successful outcome for this thesis.

Some problems and conclusions that arose from previous studies in the matter will be taken into account, however there is still opportunity to create a different approach, based on the company's current status and based on the company's objectives, as every company is peculiar in its own way.

2.1 Introduction

The issues associated with software development are not new, every company from little to huge passed through these issues, some conquered it, others are still fighting it and others have not started to fight back.

Since the beginning of the computer era that visionaries saw that managing software development was not an easy task, and many processes were created through the years. One of the most successful models was created at the Software Engineering Institute at Carnegie Mellon University in Pittsburgh, in 1987, the Capability Maturity Model (CMM), later Capability Maturity Model Integration (CMMI). This is a well-known and accepted de facto standard for assessing the maturity of software development organizations.

As the company was very interested in obtaining the CMMI Level 2 "certification", to help it conquer foreign markets, and as CMMI is a de facto standard in the software development industry, it was deeply taken into account in the research.

The initial idea of the proponent was an agile approach, specially focused on Scrum, and since it is a well-known and successful process, it was deeply researched.

Due to the outstanding results of the Team Software Process (TSP), especially regarding to quality, this process was deeply researched.

On further analyzing the company and the methodologies it was decided to also consider Kanban in the mix.

The main idea in this state-of-art review is to understand which methods gave better results to issues similar to the company's issues. And to learn from past experiences what kind of success one could expect from each of the processes and their "mixology".

2.2 Capability Maturity Model Integration – CMMI

The CMMI (SEI 2010) is known as a de facto standard across the world and in many cases, the level of maturity can mean a contract or not. Many customers give preference to higher level of CMMI maturity as it reveals that the company follows a plan-driven management system and not a crisis-driven one.

"Because crisis-driven organizations typically cannot sustain complex improvement programs, the first improvement step must be to fix the crisis-driven management system. That is why, for example, the CMMI improvement program calls for improving to level 2 before doing anything else." (Humphrey and Over 2010)

CMMI – Dev is represented by 5 levels of maturity, from 1 to 5, by 4 levels of capability, from 0 to 3, and by 22 process areas.

For achieving the maturity level 2, a company should have the capability of 2 or above in the following areas:

- Configuration Management (CM);
- Measurement and Analysis (MA);
- Project Monitoring and Control (PMC);
- Project Planning (PP);
- Process and Product Quality Assurance (PPQA);
- Requirements Management (REQM);
- Supplier Agreement Management (SAM).

By design, CMMI is a model of best practices, telling what practices should be in place, but not how they should be implemented and organized together – that's the role of concrete processes and process frameworks, as are the cases of Scrum and PSP/TSP. (Garzas and Paulk 2013) (Garzás and Paulk 2013)

As it can be said that CMMI works at a different level from the other research methodologies it was decide to include a brief description of CMMI with other methodologies just to understand their impact in the CMMI certification.

2.3 Scrum

Ken Schwaber and Jeff Sutherland first co-presented Scrum at the OOPSLA conference in 1995 (Schwaber and Sutherland 2013). Scrum is lightweight, simple to understand but difficult to master! (Schwaber and Sutherland 2013)

"SCRUM is a framework for developing and sustaining complex products. (...) consists of roles, events, artefacts, and the rules that bind them together. (...) is not a process or a technique for building products; rather, it is a framework within which you can employ various processes and techniques. (...) is founded on empirical process control theory, or empiricism. Empiricism asserts that knowledge comes from experience and making decisions based on what is known. SCRUM employs an iterative, incremental approach to optimize predictability and control risk" (Schwaber and Sutherland 2013)

Scrum, described in the Scrum Guide - (Schwaber and Sutherland 2013) is a process framework that can be used to manage complex product development. It is composed, in terms of people, by self-organizing and cross-functional teams, Scrum Master and Product Owner.

In Scrum the development is divided in Sprints, each Sprint should be of the same size and not bigger than a month. In each Sprint the goals must be clear, transparent and divided in a set of "tasks" – everything that might be needed to be done in the Sprint.

Scrum is a tool (Kniberg 2010) that should be used in favor of the organization and of its knowledge workers.

Scrum is composed by roles, ceremonies or events and artefacts. In the following topics Scrum is described in detail.

2.3.1 Roles

In Scrum there are 3 roles, the Product Owner, the Scrum Master, and the Development Team, which will be referred in this document by Team.

Product Owner

This is the person responsible for the final product with a clear view of the client's wishes, priorities and goals. His main job is to maximize the work being done by the Team, deciding what to build and when to build it, he manages the features wish list, priority and value, ordering items in the Product Backlog. He is responsible for doing Product Backlog visibility, all-team understanding, feature detailing and validation. (Teixeira 2013)

The Product Owner should have a very open and clear communication with the Team. And it's crucial that this communication happens whenever it is necessary so there are not doubts about the requirements and so that the "client" walks in pair with the development so that if the task is not moving towards the client's goal, adjustments are made in an early stage.

The Product Owner is responsible for maximizing the value of the product, and the work of the Development Team.

Scrum Master

He is responsible for adding and removing elements that may affect the Team's productivity, a problem solver, issue fixer that assures that Scrum agreed-upon theory, practices and rules are followed all times. (Schwaber and Sutherland 2013)

The Scrum Master also leads or maintains order in Scrum Meetings, helps the Product Owner finding effective Product Backlog management techniques and coaches the team in self-organization. The Scrum Master does not lead the team, he insures adherence and continuous improvement, and the team will organize itself. The Scrum Master does not have the authority to manage Scope, Cost or Risk.

In summary he should protect the team from outside interferences, remove impediments to the teams' proper function and serve as a mediator between the outside and the Development Team.

Team

This is the Development Team that should be composed of cross-functional, self-organizing individuals that build or develop the Product. Because the Product Owner and Scrum Master are not managers in the traditional sense they need to allow self-organizing and work in internal consensus. The Team is responsible for Sprint Planning, Sprint Retrospective, and continuous improvement of Scrum practices.

This type of Team is very important because knowledge workers need to be committed to their tasks, and assume compromises that they have agreed-upon. And not to be imposed unrealistic commitments.

As teams are cross-functional and self-organizing, the knowledge workers have the opportunity of picking which task to do from the board, according to priorities, which gives an empowerment to the knowledge worker that motivates him.

2.3.2 Events

Scrum is composed by 6 different types of events. All Scrum events are previously scheduled time-boxed such that every event has a maximum duration, and should happen on time. (Schwaber and Sutherland 2013)

One of the big advantages of implementing Scrum is its meetings – ceremonies. As these are ceremonies, where all the intervenients should participate and speak freely, and the script of these meetings, especially daily Scrums, obliges people to communicate.

As so, Scrum ceremonies will help to improve communication between team members and management, and hopefully between teams. They will also help reduce unscheduled meetings,

nonproductive meetings and the time wasted in those. To regulate the Scrum ceremonies, the Scrum Master and the coach roles are fundamental.

The whole idea of Scrum is transparency and the open and clear communication that should happen is these meetings should help everyone to be transparent about their work.

Sprint Planning

Each Sprint starts with a Sprint Planning meeting, where the Team discusses and agrees on a set of features/items, which they will collaboratively implement during the Sprint. These items are chosen form the Product Backlog and it is imperative that the higher priority/more valuable items are chosen.

The Product Owner presents the most valuable/high priority items and clarifies them, if necessary with the analysis support. The Team then selects how many they can fit in the Sprint. The Scrum Master maintains order and Scrum good practices focus. This accurate selection is the result of two assumptions: the sprint time-boxed size being always the same and the historical average of how much work can be delivered in that time-boxed period based on past Sprints. (Rubin 2012)

The meeting adjourns when a consensus is achieved on what will be the list of issues to implement representing the Sprint Goal.

The fact that the Team agrees with the Sprint goals makes them assume a commitment with the Product Owner and the feel of ownership of the project, motivating the team to achieve the goal. This is also a teambuilding factor, since it promotes team cohesion, because is the whole team that agrees with the commitment and not only one person assuming that commitment.

The team's capacity should be stable from the beginning to the end of the Sprint, so if any known alterations will happen this also needs to be taken into account in the planning and when analyzing the Sprints Velocity in the end of the Sprint.

As Henrik Kniberd described in his paper, "Scrum And XP From The Trenches", planning for the complete capacity of the Sprint is not a good idea, since there are always unplanned tasks that consume time, as meetings, bug fixing, and so on. As so he described the use of a Focus Factor, which represents the percentage of the time in the Sprint that will be planned. The remainder will be used for unplanned tasks.(Kniberg 2007)

Daily Scrum (Daily Standups)

In the Scrum Guide by Ken Schwaber and Jeff Sutherland, there are 3 fundamental questions that must be answered, by every Team member, during the Daily Scrum meetings.

1. What did I do yesterday that helped the Team meet the Sprint Goal?
2. What will I do today to help the Team meet the Sprint Goal?
3. Do I see any impediment that prevents me or the Team from meeting the Sprint Goal?

There are some rules that must be followed in these meetings and it is the Scrum Master's job to guaranty those are followed:

- The meeting is short, 15 min for a team of 5-7 members;
- The meeting happens with everyone standing, so people do not get too comfortable and to prevent delays;
- It needs to start on time, and small fines for delays or no-shows should be applied. The money gathered from these fines should be used for team building activities. (Kniberg 2007)

This meeting reinsures the Team's commitment to the Sprint Goal and provides a safe haven to communicate daily status. Everyone is aware of what the Team is doing to achieve the Sprint goal. The Scrum Master is responsible of making sure that all non-Daily Scrum subjects are dealt in another meeting. (Teixeira 2013)

Sprint Review (Demo)

It happens at the end of the Sprint and it is where the Team presents to the Product Owner what was done. It's basically a product demonstration.

Feedback and team collaboration are the main goals. The Product Owner acts as an inspector of the work developed accepting an implementation as done and/or providing feedback on the demonstrated features. Some of the issues are accepted as "Done" others will be needing additional work. (Schwaber and Sutherland 2013) Encountered changes will be included in the Product Backlog and planned for the next Sprint based on prioritization.

Definition of done

The definition of completed or done should be known to all the Scrum participants across the organization, so it is clear for everyone the meaning of "done". It can evolve through the Sprints but it needs to be agreed upon.(Schwaber and Sutherland 2013) The idea is that as the Team matures, this definition of "done" is a complete and virtually defect free item.

Sprint Retrospective

This meeting happens after the Sprint Review and is a self-examination. The Team will analyze their own performance regarding the Sprint execution, i.e., analyzing the project requirements execution. (Teixeira 2013) Retrospectives are extremely important in Scrum since they represent the best chance to improve.(Kniberg 2007)

This is a lessons learned meeting where the team can communicate freely, it's a safe place, where the team should celebrate success and identify improvements. In this discussion, matters that influence the product development, and not directly related with Scrum practices, like tools or stakeholder communication should also be discussed. The team, the Product Owner and the coach, if it exists, should be part of this meeting. If a coach exists he can share the lessons learned across teams and serve as a bridge. (Kniberg 2007)

To help with the celebration of accomplishments and to help identify improvements the following questions help break the ice:

- "What things should we stop doing?";
- "What should we start doing?";
- "What is working well that we should continue to do?".

The Scrum Master takes notes, maintains order and acts as a facilitator to a positive discussion.

These meetings are extremely important not only because they are lessons learned, and we should always learn from our mistakes and experiences to improve. As the fact that it is a place to celebrate accomplishments, which should always be celebrated since it is deeply motivating.

In the end all elements agree on the most priority/value improvements, which will be transformed into actions and implemented so that their performance is revised on the next Sprint Review. Again this requires the participation of every member of the team. (Schwaber and Sutherland 2013)

The development teams should be encouraged to point out process related problems and suggest solutions for resolving them during the retrospectives. Their suggestions and feedback needs to be taken seriously and acted upon by the managers. This will empower the knowledge workers to be more outgoing and share their feedback and complains more openly, resulting in more motivation. (Nikitina, Kajko-Mattsson, and Strale 2012)

Only with this type of clear and open communication we will be able to evolve and grow, only the ones using the process can really give feedback about what works and what does not and they need to be heard.

If there are alterations to be made we need to prioritize them and choose only one or two to settle in the next Sprint. We should also identify volunteers within the team who would be responsible for the alterations implementation. Choosing only a few process problems will help the team to stay more focused, structured and efficient. (Nikitina, Kajko-Mattsson, and Strale 2012)

Grooming

Product Backlog is a living artifact and its value is linked to its detail, organization, prioritization and size so there are sufficient items to feed the next Sprints. The Product Backlog needs to be in constant revision, Grooming activities, which may occur in the project kick-off, but should be maintained throughout the Sprint cycle and they are divided in 3 groups (Rubin 2012):

- Creating and refining Product Backlog Items;
- Estimating Product Backlog Items;
- Prioritizing Product Backlog Items.

This is a collaborative activity that should be done by the Team, so requirements are clear and it is the Product Owner responsibility, since the customer value drives the refinement process. This continuous refinement should focus the high priority items.(Teixeira 2013)

Sprint

The Sprint represents a time-boxed iteration, which is planned, executed and completed focusing the value of the deliverables to the client. It can happen that a complete feature does not fit in one Sprint. However it can be divided into subtasks, that even if not deliverable they can be considered as "done", according to the definition of "done" accepted by the Team and by the Product Owner.

At the beginning of every Sprint the Team makes a commitment of delivering the Sprint Goal, defined in the Sprint Planning.

Sprints should always, imperatively, have the same length in order to create a consistent Sprint velocity, and the golden rule is that its duration is between two to four weeks, never more than 4 weeks (Teixeira 2013), and should have a well-defined, clear and transparent goal. "To be effective goals must motivate action." (Humphrey and Over 2010)

The functionalities or other requirements planned for a Sprint, must be usable, completed and potentially releasable.

As was suggested in the paper (Kniberg 2007) Teams should never plan for the Sprint's full capacity, only for a percentage. This is a good practice, because the truth is that knowledge workers do not spend 40h of the week in planned tasks (274)(Humphrey and Over 2010), knowledge workers spend time in other unplanned tasks like reading email, meetings, communicating with other knowledge workers, answering or asking questions, and so on. As so if we plan for the full capacity, it is very likely that we miss the goal. In the paper (Kniberg 2007) it is referred the usage of a Focus Factor.

The Focus Factor represents the percentage of the Sprint capacity that the Team plans, the remainder will be used for unplanned tasks.

In the paper (Kniberg 2007) is referred that a Focus Factor higher than 70% is not a good option, which in a very light observation might be accurate. It is important to understand the organization and its knowledge workers as well as to try and see what best works for each Team and each situation. More than 70% might not be advised, but in some Teams might be the right Focus Factor, agile is all about inspect and adapt.

During a Sprint nor the goal nor the quality must be endangered.

2.3.3 Artefacts

Product Backlog

This is something like the client wish list, it is prioritized by the client and it is the single source of requirements for additional changes to be made to the product.(Schwaber and Sutherland 2013)

- Content is always prioritized;
- It's a living artefact in constant change/update;
- It's the Products Owner responsibility.

Sprint Backlog

It represents the Product Backlog items selected for the Sprint, which are selected by the Team and represent what the Team committed to deliver as the Sprint Goal. The selection should be made based on previous estimations and past performance. It should be detailed enough so that changes in progress can be understood in the Daily Scrum, if new work is required, the Team adds it to the Sprint Backlog. Only the Team can change the Sprint Backlog during a Sprint. (Schwaber e Sutherland 2013)

"Only the Development Team can change its Sprint Backlog during a Sprint. The Sprint Backlog is a highly visible, real-time picture of the work that the Development Team plans to accomplish during the Sprint, and it belongs solely to the Development Team." (Schwaber e Sutherland 2013)

The task board represents the visibility of the Sprint Backlog.

Burndown Chart

It is the always visible Sprint tracking mechanism, a simple chart in which Sprint progress is measured against the Sprint commitments. This helps the team to track daily progress until Sprint completion. It should be in a type of unit universally understandable by the team, and it gives some sort of completion forecast calculated in line with the progress of the work being done by the Team.

This simple chart represents the work/effort that supposedly the team still needs to accomplish.

It can be printed and placed on/near the board to have an idea about the progress and forecast, it is reviewed in the Daily Scrum.

The chart will contain only the tasks agreed-upon in the Sprint Planning by the Team, meaning the ones present in the Sprint Backlog.

2.3.4 Estimates

In Scrum estimates are made by the team. Each team member chooses a card, or writes in a paper the effort he/she thinks that task will take, then everyone shares their estimates and only when a consensus is encountered and everyone in the Team is comfortable with final voting value the estimation is set. This is usually done in the Sprint Planning or Grooming Meeting.

In the beginning of Scrum teams did estimates using effort, or story points, which are not directly correlated to time, and it started to be done using a Fibonnaci "scale", choosing an effort from [1, 2, 3, 5, 8, 13, 21].

However, as Kniberg described in his paper (Kniberg 2007), the fact of using such a defined scale could misguide the Product Owner and the Team, giving a false sense of accuracy. *"If a story is estimated at approximately 20 story points, it is not relevant to discuss whether it should be 20 or 18 or 21."* (Kniberg 2007) As so the estimation method used in Scrum has been evolving throughout the years and one of the common methods used is Planning Poker. (Kniberg 2007):

To estimate using Planning Poker each Team member has a set of cards – see Figure 1, numbered as in the picture, and for each item, each Team member chooses a card, and places it, face down on the table. When everyone has placed their choice, the cards are turned and the Team discusses the estimates till a consensus is found. *"If you ask the team to provide an estimate, normally the person who understands the story best will be the first one to blurt one out. Unfortunately, this will strongly affect everybody else's estimates."* (Kniberg 2007) Planning Poker helps to avoid this.

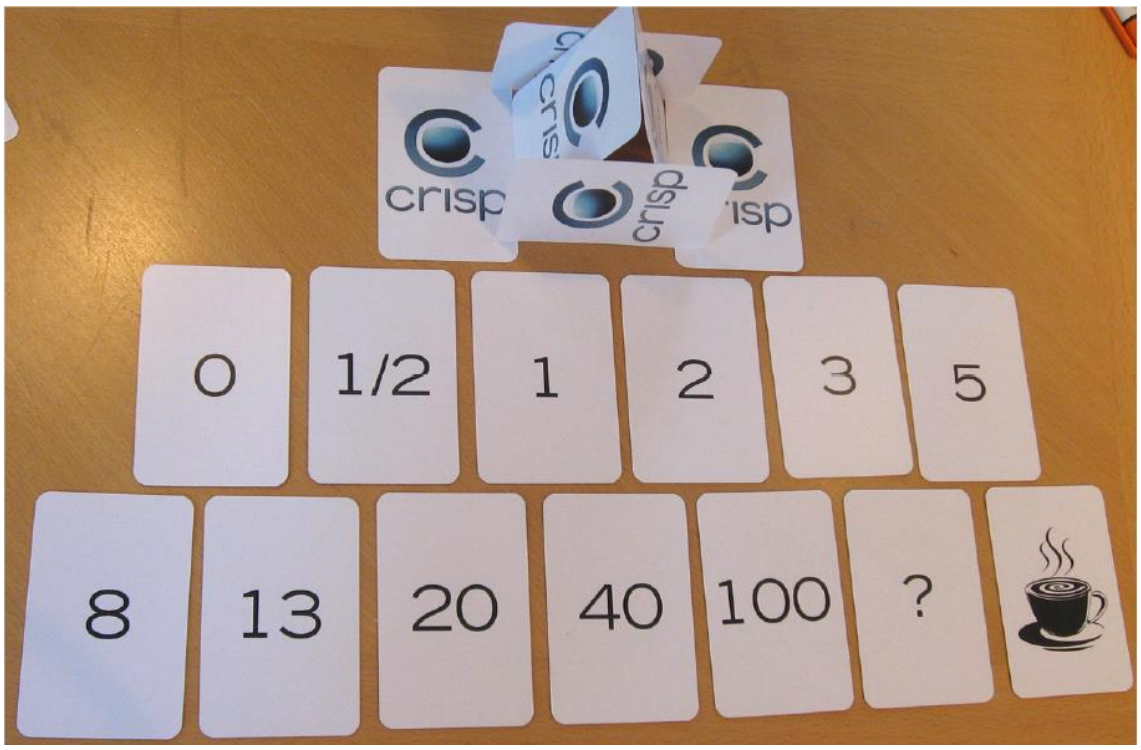

Figure 1 - Planning Poker cards (Kniberg 2007)

This set of cards has some particularities:

- Numbers like 20, 40 and 100 to avoid a false sense of accuracy for large time estimates;
- 0 = "this story is already done" or "this story is pretty much nothing, just a few minutes of work
- ? = "I have absolutely no idea at all. None.";
- Coffee cup = "I'm too tired to think. Let's take a short break.".

To help the accuracy of estimates it is always a good idea to split the stories/items into smaller ones. "And no, you cannot cheat by combining a 5 and a 2 to make a 7. You have to choose either 5 or 8, there is no 7." (Kniberg 2007)

2.3.5 Scrum and CMMI

Although CMMI and Scrum were created for different purposes they can complement each other. In the article "A case study of software process improvement with CMMI-DEV and Scrum in Spanish companies" a study was made in 12 Spanish companies to answer the question "Is Scrum useful and practical for carrying out software process improvement efforts with models such as CMMI-DEV?" (Garzas and Paulk 2013)

"The results obtained show that most of the process areas of CMMI-DEV level 2 had been improved by using Scrum. Other issues detected arose during the formal appraisals and illustrated how it is possible to verify with Scrum that the specific goals of CMMI-DEV have been implemented."(Garzas and Paulk 2013)

The article "Implementing Scrum (Agile) and CMMI Together" goes in deep detail about the merge between CMMI level 2 and Scrum, and concludes that Scrum is a good implementation for some of the practices in CMMI level 2, therefore they can be used together. (Potter and Sakry 2011)

An interesting report in this article, is that although CMMI requires evidences and Scrum represents the agility of bureaucracy free development, one simple form to gather evidences from Scrum is taking a picture of the board.

"Even for groups that like to use white boards, you can be creative and at least establish some basic protection by labeling items (e.g. "V1.1," or "Story dated 1/2/YY") and taking a photo." (Potter and Sakry 2011)

2.4 Kanban board

In software maintenance it is very common that teams receive new requests frequently, and as so it is important that we try to reduce the necessity of task-switching and multitasking. In knowledge work, task-switching can be highly unproductive (Speier, Valacich, and Vessey

1997) and however Scrum already puts in place some measures to reduce task-switching, as making the teams more goal oriented, the Kanban board attacks task-switching and multitasking more aggressively.

The Scrum board is viewed by the community as one of the most useful artifacts of Scrum (Teixeira 2013), and it was reported in several studies (Kniberg 2011) (Nikitina, Kajko-Mattsson, and Strale 2012) (Nikitina and Kajko-Mattsson 2011) (Rutherford et al. 2010) (Kniberg 2007) the impact that a physical board has compared to a virtual board, and as so my first recommendation is the use of a physical board to represent the team's work. In said papers is even referred how teams changed from virtual boards to physical boards.

When implementing Kanban, it is crucial that the columns of the board correspond to the Teams workflow of doing items, and include all the important and relevant steps that exist in their workflow.

Kanban establishes the idea of capacity, and limiting the Work In Progress (WIP) to reduce, task-switching and partially done work. As Michikazu Tanaka said *"by reducing work-in-process we motivate people to do better than they ever thought they could do, resulting in a raise in productivity and illuminating problems. By limiting the WIP we also reduce multi-tasking and task-switching, resulting in more Work Items being stated and ended without interruptions"*. (Teixeira 2013) There is no golden rule for the capacity of each column, it is another inspect and adapt point.

All the items in the board must be clear, and visible. We should be able to, just by looking at the board, clearly see who is working on that item, what is the item next stage, if it has a deadline, if it has impediments and so on.

White boards are great for Kanban boards – see Figure 2. They allow teams to draw and erase lines, so they can understand which are the columns that work for them, and only after teams have matured the board enough they can move to duke tape to identify columns. It is advised that the glue on the sticky notes is enforced by using adhesive tape. Another interesting practice is the usage of personalized magnets to identify the knowledge worker working on that item. (Kniberg 2011)

Figure 2 - Examples of Kanban boards, some with drawn lines others, more "mature" ones with black tape (Kniberg 2011)

It is extremely important that if an item has a deadline that it is represented clearly on the board. And that all impediments are visible, like per example, using pink sticky notes to identify impediments clearly. (Kniberg 2011)

The board is something that really brings value, not only because it is a clear view of the project, as of the work-in-progress, as well of all changes in prioritization that should be accounted for in the board. It offers better transparency of the development process and shows which developer is working on which task. (Nikitina and Kajko-Mattsson 2011)

There are many particularities regarding the Kanban Board, which are detailed in the following topics.

2.4.1 Visualization of team work

The board provides visualization on the stage of development of each of the issues. This is very important not only to the team as it is to the surrounding teams.

The idea of the Kanban board is that we can easily perceive in which stage a ticket is, if it is blocked, its priority, which will be its next stage, or even which developer is working on that ticket.

A known practice to identify the person working in that ticket is with different magnets. (Kniberg 2011)

2.4.2 Limit the Work In Process (WIP)

This is a key point in Kanban and a necessity to reduce the multitasking and task-switching activities that can be highly unproductive in knowledge work. (Czerwinski, Horvitz, and Wilhite 2004)

Limiting the WIP to match the team's capacity helps ensure teams do not pile up partially completed work. Adding things to WIP without completing anything just increases the duration of all tasks in the queue. WIP limits can be used to limit capacity for both human and not-human resources. (Teixeira 2013)

Multitasking and task-switching is not efficient. And to pile up tasks without limiting WIP increases the duration of the tasks, hence increasing the lead time, which significantly affects productivity and can, consequently, affect the company's profitability! (Salvucci, Taatgen, and Borst 2009)

2.4.3 Only start new work when an existing work is complete

Only start new work when an existing work is complete, this is the golden rule of Kanban and is actually a key for success and quality work and even motivation, since the knowledge worker will feel that he achieved something and did not left a quantity of tasks in half.

Capacity must be respected always, to reduce multitasking and task-switching and even to help control stress levels. By having a limited WIP we are not saying that knowledge workers will "slack", we are guarantying that in each task the focus is on that task and with that reduce the error proneness' of multitasking and task-switching. Quality needs to be the focus of any mission-critical software and this type of measures will help.

2.4.4 Kanban Tickets

Tickets represent small tasks, preferably independent, more easily manageable.

To easily understand the priorities of the tickets on the board, they can be of different colors, shapes, or even be organized in swim lanes – the options are endless.

With time and experience teams evolve their board finding the solution that best fits them. As I stated before, agile is based on continuous improvement and we should never forget to question and see if what we are doing is what works for us. (Kniberg 2011)

It's extremely important that just by looking to the board we understand the criticality/priority of each issue so that issue is chosen for the next phase.

The fact that tickets are organized by priorities brings commitment to the knowledge workers, because if they have many tasks with the same level of priority, they can chose the one they prefer to work on. And nothing is more motivating than doing what you like.

2.4.5 Pull system

In the board, at the bottom of each column should be the definition of done to that particular column (Kniberg 2011). For example, in the test column it could be: "All the defined test cases where executed".

Pulling a ticket to a new column represents that that ticket was done, according to the definition of the previous column.

As this is a pull system, and we have a limited WIP we may encounter, as the system matures, some bottlenecks, this meaning that the previous column has completed items but the next column has not free capacity to receive those items. It's basically a metric of the Kanban system capacity, and a performance completion indicator. (Budau and Bihler 2012)

This is another great advantage of Kanban, the visualization of bottlenecks to see when a team or a team member needs a hand. (Budau and Bihler 2012)

2.4.6 Cycle-time and Lead-time

Cycle-time is a time measure, starting when a work item begins to be implemented (pulled from backlog) until being considered done (when it reaches "Done" column).

Lead-time is also a time measure, measuring the time from the customer request creation to the delivery of such request. It's a customer request response time, a metric that gives the teams the customer's own vision, and can be used to forecast future response times and give timeframe delivery commitments. Lead-time includes the Cycle-time and is basically a request arrival indicator. (Teixeira 2013)

2.4.7 Class of service

Each class of service has its own set of policies that affect prioritization decisions. There are several classes of service, but as a general guideline, more than 6 classes would probably become too complicated to administer and operate. The number of classes should be small

enough that everyone involved can remember its meaning and sufficient enough to offer flexibility. (Anderson 2010)

As an example, 4 types of class of service will be detailed bellow:

- Expedite (or "Silver Bullet") corresponds to a ticket that needs immediate response. It can have a different workflow in order to jump some of the queues or reassign some resources, or even ignore WIP limits.
- Fixed Delivery Date – This is a ticket with a deadline, and as so everything must be done to ensure that the item is prioritized into and through the Kanban system in sufficient time – not too early but never late – Just-In-Time.
- Standard Class – standard ticket, the more common class of service, with an immediate delay cost and lesser customer urgency.
- Intangible Class – This is a ticket that cannot be converted to a tangible business value but the business either knows intuitively to be important or has some subjective empirical evidence such as customer complaints on which to base a valuation. Intangible items may include production bug fix requests, usability improvements, branding and design changes and their like.

Classes of service are only advised in mature Kanban teams and are usually represented by tickets of different colors.

2.5 Scrum and Kanban

Scrum and Kanban is not an unusual combination (Kniberg 2010).

Kanban just by it is not enough. In articles like these (Rutherford et al. 2010), (Nikitina, Kajko-Mattsson, and Strale 2012), (Nikitina, Kajko-Mattsson, and Strale 2012), (Kniberg 2011), (Nikitina and Kajko-Mattsson 2011) we see that when adopting Kanban companies also use some parts of Scrum, usually the daily meetings and the retrospectives.

Scrum and Kanban complement each other, Scrum has its time-boxed meetings and its oriented agility of coherent Sprints, and Kanban has the completeness of its board, which allows teams more agility to deal with software maintenance.

Self-organizing and Cross-functional teams

For both Kanban and Scrum, self-organization and cross-functional skills are a success factor, being self-organizing the key. (Teixeira 2013) Teams work and adapt as a whole in order to accomplish the items, without the need of management supervision or intervention. "It's a power-to-the-people continuous improvement concept." (Teixeira 2013)

When teams are cross-functional, any member has the skills needed to pick an item and complete it, without depending on other members, which means that regardless of the absence of team members, every work item can be completed.

In the real word this might not be easy to accomplish, as team members might not be interested in learning new things and stepping out of their comfort zone, however this is a success factor that should be encouraged at all times.

In Scrum, the need of cross-functional teams is even higher since the team commits on delivering a bucket of items, and they cannot rely on individual team members to accomplish the goal. Not forgetting of course, that with cross-functional teams, estimations will certainly be more accurate.

2.6 Team Software Process – TSP

One of the reasons that make many companies to adopt TSP is that it can help achieve significant improvements in product quality, reduced development time and more accurate project estimates. (Koch 2005) Because to really improve something you need to manage it, and to manage it you need to measure it. The foothold of PSP/TSP is to gather accurate and complete data to do high-quality work. (Humphrey and Over 2010)

The problem with PSP/TSP is that many companies are afraid to really invest in it, as it means a big initial commitment in time and money, although this commitment can be rapidly recovered in the first TSP project.

Companies that adopt TSP report a tenfold reduction in mean schedule error, reduced test schedules and a 30% increase in functionality. Adobe and Oracle reported improvements in predictability, productivity, and quality. (Humphrey and Over 2010)

"Most organizations report productivity gains of at least 25% to 30%, from requirements through user acceptance test, and similar gains for design through system test or user acceptance test." (Humphrey and Over 2010)

"The quality management system in the TSP is so effective that most organizations are reducing defect density by 70% to 80% or better while also improving productivity." (Humphrey and Over 2010)

"The most efficient way to do a job is to do it right the first time."(Humphrey and Over 2010) This is the core idea of TSP, high-quality work. That is why well known organizations like Microsoft's IT Organization (Grojean 2005), Adobe, Oracle, and so on, use PSP/TSP.

From reading these enthusiastic reports we cannot wonder why everyone is not using PSP/TSP. The thing is that beside the time spent in training developers through PSP and TSP, training the managers with TSP, having certified coaches from Software Engineering Institute (SEI), or have workers getting certification is very expensive.

So companies are afraid to invest time and money certifying their members, due, e.g., to their market position, company's financial situation or workers turnover. These are, from what I understood, the main reasons why not all companies adopt TSP from the start.

But the success of TSP is highly dependent on the commitment of the members in the company. Workers need to feel the management support to keep gathering data, and doing the code reviews, even when the work gets crazy, otherwise all the efforts in implementing TSP are wasted. This is also valid for middle management that needs the support of higher management.

"The TSP is implemented with a combined top-down and bottom-up strategy. That is, the first step is to get top-level executive support and to produce an overall organizational plan. Then you start actual implementation at the bottom level with individual TSP teams." (Humphrey and Over 2010)

Another big issue with TSP is trust, trust is very important, especially in the TSP teams, management should trust team members in their proposals of commitment but as W. Edwards Deming said "In God we trust; all others must bring data", it cannot be a blind sighted trust. In the TSP launch, teams should provide accurate data demonstrating why they are committing to that schedule, and it is the management job to verify that the data is correct and it is applicable to that project, this meaning that was compared with similar jobs/tasks and that it was accurately gathered.

"TSP has brought credibility to estimates and commitments to perform worthwhile in a maintenance environment," "We are no longer questioned on the basis of estimating our requirements because we have the data and the performance to back it up." said AV-8B JSSA's IPT Lead Dwayne Heinsma (Rickets 2005).

"From a management perspective, it is essential that software estimates used in a TSP launch are as accurate as possible. Significant growth due to estimation inaccuracy can wreak havoc on a team attempting to stay within cost and schedule while executing its established plan." (Sinclair, Rickets, and Hodgins 2011)

In the study "A study on how software engineering supports projects management", involving CMMI, ISO 15504, Unified development Process (UP), Xtreme Programing (XP), PSP and TSP, TSP was considered:

- One of the most usable methods, because the expert's technique of multiple voting is more attractive to practitioners;
- One of the easier methods to estimate software projects;
- Provides the meetings required, the agenda, roles, recommendations and examples;
- One of the most usable methods because the process, based on meetings is very attractive to practitioners;
- Analysts, designers and programmers considered TSP one of the easier due to the guidelines that enable their participation in the planning activities (Sanchez-Segura et al. 2008)

TSP and CMMI

Another interesting fact, reported in the article "TSP can be the building blocks for CMMI" is that TSP can potentially reduce the time and effort required to achieve your CMMI goals, eliminating the need to choose between two sets of laudable objectives.

"Hill Air Force Base reported achieving CMM Level 2 in a relatively quick 14 months by adding the TSP to their strategy. Then, in January 2004, the base made a follow-up report about moving from CMM Level 2 to Level 4 in only 16 months (as opposed to the normal 50 months), crediting the TSP with their almost unheard-of pace. Hill AFB's experience shows that you can capitalize on the TSP's proven framework for process improvement to speed your CMMI initiative along." (Koch 2005)

This is a very interesting point since the company wants to achieve CMMI level 2 certification by the end of 2014.

2.7 Organizational Change Management - OCM

Organizational change is a continuous and arduous process that requires the involvement of all the personnel, and the management sponsorship is crucial for its success.

"Those organizations where change is attempted usually fail in their efforts (66% according to one estimate) or achieve only marginal effects." (Sturdy and Grey 2003)

"Building support and understanding from the top down reduces risk and resistance to change." (Humphrey and Over 2010)

Process improvement it is not simply a technical problem, the social aspect is probably the primary factor of every change effort. Cultural barrages can significantly hinder a software process improvement program or prevent it from succeeding. (Kandt 2002)

To access the difficulties inherent to change, especially in organizations, there are four key principles underlying organizational change best practices, reported in the paper "Organizational Change Management Principles and Practices" by Ronald Kirk Kandt (Kandt 2002).

Principle 1: Business processes must support business needs – Change should be in the best interest of the organization and its objectives. Change should be part of the road that leads the organization to its goal and vision.

Principle 2: Staff an organization with people that can successfully support and execute the business processes – Management is critical to support a change effort, they are the responsible ones to keep the other motivated to change.

Principle 3: Plan change efforts to maximize return on investment and minimize risk. – Nothing like a detailed plan describing the benefits, the barriers and the alternatives for infusing change within an organization.

Principle 4: Measure process and product quality. – This measurement is extremely important, not only to keep track on the change, but also to evaluate the success of the change iteratively.

There are twenty-four practices that support the 4 principles, and these fall within 4 groups. The practices and the groups are detailed in the next topics.

2.7.1 Establishment Practices

- Practice 1: Align the goals of a change effort with organizational strategy – the goals of the change effort must support the organizational strategy;
- Practice 2: Acquire and maintain executive commitment – great leadership is crucial to the success of change, executive commitment will influence the behavior of the entire organization and make it more willing to adopt change;
- Practice 3: Create and maintain a superior change team – this team must have some degree of authority and its members should be open-minded and driven, have good management and communication skills and have the ability to focus on the change vision and objectives;
- Practice 4: Evaluate the willingness of the organization to change – failed change will increase the challenge, and the effort, this is why planning it is so important;
- Practice 5: Change teams must be the instruments of change – Senior line managers should be sponsors of change, high rank members should be part of the team and the team should periodically communicate with those affected by change;
- Practice 6: Plan for continuous improvement – use good performance methods and monitor and respond to them.

2.7.2 Execution Practices

- Practice 7: Articulate an extremely compelling need for change – To do this an organization should first access the current status, and then compare it with the desired one, so it is clear for everyone the importance of the change;
- Practice 8: Select processes to change based on those having the greatest expected return on investment and the lowest expected risk – There are four kinds of processes of strategic importance, the most important processes for an organization are the identity process – purpose of the organization – and the priority process – related to what the organization produces;
- Practice 9: Change at most three processes during a change effort – to maximize return on investment and minimize risk organizations should adopt a strategy of

gradual change characterized by several minor changes so its personnel can better cope with the change effort;

- Practice 10: Create a vision for each process to be changed – The process vision must describe the new capabilities of the process and the expected performance improvements, with measurable objectives;
- Practice 11: Develop an "as-is" understanding of the processes to be changed – A change team must understand existing processes to identify areas of improvement, to estimate how much an organization can improve, and to measure improvement;
- Practice 12: Understand the risks and develop contingency plans – the best way to mitigate risk is to implement change through pilot efforts that demonstrate success;
- Practice13: Follow software assessments and project postmortems with planned process improvement programs that eliminate or minimize noted problems – Two benefits: reinforces management's commitment to improvement; and helps to improve product quality and personnel productivity.

2.7.3 Monitoring Practices

- Practice 14: Select and use appropriate metrics – measure the desired characteristics of a change effort;
- Practice 15: Perform annual process assessments and benchmarks;
- Practice 16: Continually measure the productivity of personnel and the quality of software artifacts – Continual productivity and quality measurements help to identify organizational problems, whether they are process, cultural, or motivational in nature;
- Practice 17: Analyze an organization's software portfolio, which is the total number of applications it owns – an organization should know the role each application plays in the future of the enterprise and the areas that have the greatest need for tool support;
- Practice 18: Conduct postmortems of software projects – to analyze the strengths and weaknesses of the defined software processes.

2.7.4 General Practices

- Practice 19: Install in the organization a commitment to change – management should be committed to change, and the organization should make an effort to recruit resilient people, which are positive, focused, flexible, organized, and proactive. The proposed change should also be consistent with past changes;
- Practice 20: Communicate effectively – Communication is the most effective tool to obtain acceptance of change and it should be face-to-face to demonstrate the

necessary commitment. *"(...)effective communication will help the workforce better understand what the change is, the motivation for the change, and how it affects them"* (Kandt 2002);

- Practice 21: Top-level executives should stay actively involved – they should participate in all the stages of change, and express their commitment. Their support is an extremely important part of change's success.
- Practice 22: Listen to the customer – customer should be involved so its strengths and weaknesses are understood, and its desires and needs satisfied;
- Practice 23: Align the infrastructure – organization's business units should be aligned, and if personnel needs to assume new functions they should have the proper training;
- Practice 24: Foster a creative and innovative environment – The search for new ideas and ways of doing things better should be constant. The new ideas should receive the proper recognition, for better or for worst.

The most critical element for change's success is the energy and momentum of the implementation effort itself. *"To build and sustain this energy, you must name a hard-driving and enthusiastic change champion, form a strong implementation team, and establish and follow an aggressive implementation plan."* (Humphrey and Over 2010)

2.8 Summary and Conclusions

As stated above CMMI and Scrum can complement each other and TSP can build blocks for CMMI. And Kanban as the core of lean, simply means creating more value for customers with fewer resources.

The article "Management challenges to implementing agile processes in traditional development organizations" makes some good points in the merge of agile methods and classical ones. (Boehm and Turner 2005)

"Measurement is critical in agile methods, particularly in establishing development velocity values critical to time boxing. (...) Quality and defect management are also intrinsic to most agile methods, but the activities don't necessarily match one to one with some of the traditional approaches." (Boehm and Turner 2005)

The more critical barriers found are perceptual and not technical, lessons learned can support more rapid integration, this is why the state-of-art review is so important in this project.

It can be concluded, by the review of the state-of-art, that this four well known referential can work together. Scrum and CMMI complement each other in some aspects, TSP brings the quality focus, that is crucial to a company's success, and may provide significant reductions on test and quality costs, and Kanban is suited for managing continuous maintenance work.

Management and other teams in direct connection to the pilot teams, like quality assurance, must sponsor the project. It is also very important that the team members selected to the project are volunteers and early adopters who are eager to be part of the pilot program. It is very important that skeptics do not participate in the pilot program. (Humphrey and Over 2010)

Chapter 3

Diagnosis

The success of any organization depends on its ability to respond quickly to an ever changing market and hungry customers.

The goal of this project is to help the teams in a mission-critical software company, to optimize their software development process to achieve better results, focusing on quality. As this is a mission-critical software company a defect may, indirectly, result in the loss of life, so quality is indeed a critical factor to this company's success.

The first stage of this project was to analyze the current system and work method, this stage is the diagnosis. This diagnosis was the foothold of the proposal that was implemented in pilot teams

3.1 Introduction

The complete diagnosis was presented to the company, however in this chapter I will only make a short exposé of what was identified that is within the scope of this dissertation.

The success and failure of projects has been analyzed over the years. One good example of this is the CHAOS Report (Group 2014), which has been created biannually for the past 15 years by the Standish Group.

Conclusions regarding the success and failure of projects were taken not only from studies but also from surveys done at IT companies. A mapping of those success or failure factors based on "Agile and Traditional Project Management: bridge between two worlds to manage IT Projects" (Teixeira 2013) to topics of this diagnosis is detailed below:

- "Stakeholder Involvement" is one of the primary influences for product success, and the lack of involvement is a strong failure factor, this is also related to "Communication and Collaboration" – see Requirements;
- Lack of "Communication and Collaboration" is the top reason for failure – see Communication and Collaboration;
- "Organization Support" is the second top point for success and the lack of such support is also visible as being a big issue failure-wise. This is related with Communication and Collaboration as well as other factors like motivation, trust, support, that are referred to in many topics in this paper but especially in the topics Trust, Motivation;
- "Skills & Knowledge" is the less important factor for failure but it comes in second when stating the factors which are needed to ensure project success – this relates especially to the Training, Transversal Resources, and Cross-functional Resources .

"Fostering the skill, creativity, and motivation of your knowledge workers and forging them into a productive workforce must be your highest business priority." (Humphrey and Over 2010)

"High-quality work is not done by mistake, and it certainly is not done by people that don't care." (Humphrey and Over 2010)

With this in mind I would like to report the importance of knowledge workers' motivation and this will be one of the key points of the diagnosis.

This chapter is composed by a first topic that describes how the interviews were done, followed by the topics Communication and Collaboration, Team Management, Project Management, and Quality Management, ending with some brief conclusions of the Diagnosis.

3.2 Interviews "How to"

To undertake an accurate proposal on methodologies that help improve productivity and collaborators' satisfaction, it is important to understand the current status of the company and conduct a lessons learned analysis on past experiences, growth, culture, nature, and DNA of the organization. For that nothing better than observing and openly communicating with the knowledge workers.

To do this I performed a series of confidential interviews so I could have a view of the company's "now" status, from very different angles. I interviewed the developers of all development areas including designers, analysts and testers. I also interviewed all the Senior Product Managers (SPMs). This represented 70 interviews.

To do these interviews I used a questionnaire created by Dr. Luís Graça[2] – see Appendix A, which tries to encounter our role in a team. The questionnaires were made available to the knowledge workers with time so when a knowledge worker stepped to the interviews room I would have his questionnaire already analyzed. This would serve as a conversation ice breaker so the interviewee was more comfortable with the interview.

This questionnaire was also something very positive because in the weeks that followed the interviews there was a hubbub on the floor, people sharing the role they got and in the majority of the cases they felt it applied to them.

The interview would start with the interviewee explaining in which points the role(s) that he got were adequate, or with which parts he identifies, and then I proceed with the interview. So that the interviews were performed under the same conditions for all the participants, a script was followed.

These interviews help to understand the key aspects of software development at the organization, as well as the collaborators satisfaction with the development process implemented at the moment.

Due to the restriction on size of this document the script of the interview can be observed in Appendix A - Interviews; to each question we will see an explanation on the rational of the question. When interviewing the middle management, as it was in the end of the interviews process, I introduce some extra and more precise questions that arose from the previous interviews.

With these interviews I've created an extensive document with the most relevant sentences pointed out by the knowledge workers. This compilation will not be part of this document as it is highly confidential.

When compiling all of this information, I decided that it was not enough because there were some discrepancies in information and even misinformation. Hence, I went further and decided that it would be important, and even crucial, for my diagnosis and to better understand the company's view, to include other research areas.

Therefore, I decided to include more elements in the interviews, like key elements from the Operations team, the HR Manager, one of the Department Directors and the IT Manager. This reinforces the idea of the importance of talking not only with the Development teams but also with other agents that connect with Development workers.

These interviews were very different from the others in script, because my interest was to expose issues found in the previous interviews and attest to their veracity. It was very important since they allowed me to understand both sides of the coin and conduct a more detailed and correct diagnosis.

There are some other members that would be interesting to interview, but we needed to draw the line at some point due to the dissertation time-frame.

[2] The original questionnaire can be found here: http://paginas.fe.up.pt/~ei07157/assets/eu_tu_equipa.pdf

The results from the interviews, the insight gathered from observing knowledge workers at work combined with the knowledge acquired from the state-of-art review, resulted in the diagnosis of the company's "now status".

This diagnosis is one of most important outputs in my dissertation. It is a compilation and analysis of everything that makes this company, its culture, its DNA and most importantly its people. The following topics detail, in a generic form, the points of the diagnosis relevant to this dissertation.

3.3 Communication and Collaboration

3.3.1 Communication

Communication is an important aspect that influences the development process and especially knowledge workers.

Clear and open communication can represent a success factor in every aspect of software development. From good communication with the client to good communication with the management, so orders and vision are well understood, and ending in good communication between knowledge workers, all of which result in a better work environment and higher product quality.

Communication is a problem in the organization not only inside teams, as between teams, as with management. Another important aspect about communication is the lack of feedback that some knowledge workers at the organization have. Knowledge workers need to feel that their work is noticed and feedback, be it positive or negative, is usually a motivational factor that needs a more regular output.

"almost all knowledge workers, when properly motivated and challenged, are highly creative and productive." (Humphrey and Over 2010)

As a big company, there are too many channels to share information and to communicate. This is to say, information is often lost and the result is frequent misunderstandings.

Another problem with communication is that it is deeply related with the documentation. The company's documentation, especially regarding the development process, is peculiar. To better understand the issues inherent to documentation, please refer to the Documentation section.

The communication needs improvement in many aspects, not only from management to the knowledge workers as between knowledge workers. Teams do not communicate enough between them, do not share ideas, or difficulties, or even developments that will affect other teams.

3.3.2 Documentation

Documentation is one of the most crucial areas in need of improvement and it is very clear, not only from the interviews that I conducted, but also from my experience in trying to find documents at the company. There are a substantial amount of problematic issues that are partly provoked by the lack of proper documentation.

Documents are created and "abandoned" in a Windows folder system, with no real search functionality, instead of organizing the information in the wiki, which the company already has, but that is completely outdated in almost all the fields.

At this moment if you want to understand how a feature works, you will need to find all the documents of all the versions where that feature was altered. This represents and enormous investment in time, which teams, especially Operations, usually skip, and ask directly to the developers. The product is so complex and there are so many possible customizations that this difficulty with documentation is a huge obstacle. It exists a big necessity of constant communication, especially between the Development and Operations teams because the Operations team cannot easily find documentation, or because it does not exist, or because it is outdated and sometimes because it is easier/faster to ask directly to a developer.

The fact that the documentation is so disperse, in Windows folders, which difficults the job of everyone, and especially of the functional analysts when they try to do an impact analysis.

Much of the know-how is on the heads of the knowledge workers, and this is a huge failure factor for the company. An organization should not be so dependent on the particular individuals that work there.

Documentation should never be a burden, it should be seen as support, and the importance of accurate and complete documentation should be clear to everyone, especially on an organization of this size.

The passing on of knowledge is a major problem within company's development area at the moment because there is not enough documentation to support it. So it is crucial that the know-how and knowledge does not remain only with individual employees, but is duly put on paper. The concentration of knowledge in specific human resources is a serious failure factor.

Some of the teams are starting to write FAQs or functionality description documents so it is easier to communicate with the Support and Operations teams. Just by this we can understand how the developers feel about the need for more documentation.

Of course the organization needs agility in many aspects, but agility does not mean losing focus. Furthermore, too much agility is not functional when so many people need each other. There is such a thing as "too much agility". This is a company that has a very complex product and many people work for it, so it is important that some degree of organization exists in order to obtain success.

In an organization so big as this, clear and easy to find documentation is a success factor, of course guarantying the confidentiality needed in an organization of this size.

3.3.3 Summary

Communication needs to be improved and that will represent an exponential effort from everyone. Knowledge workers will need to have more room for new ideas to blossom and they need to be listened to, because they are the ones that best know their job.

Knowledge needs to be shared, inside the team and to other teams. The organization should not be dependent of single individuals.

Relating to documentation, there is much that can be done. These are just some of the improvements that can be done:

- Centralized information;
- Searchable information;
- Easy to find information;
- Easy to browse information;
- Information that is accessible to knowledge workers;
- Information that is editable by the needed persons (in some cases all team members should be able to edit and in other cases editing should be restricted);
- Reliable information which is validated by team leaders;
- Information that is easy to read, especially for FAQs and "How to's".

3.4 Team Management

Constant change is not only a well-known fact in the software industry as it is in the company's DNA; we cannot simply run away from it because it will always chase us, so we need to embrace this reality and make the most of it. This is why agile is such an important approach.

The "official" development process is "waterfallish". However within the organization, the process is often left behind and many times it is not even put into action. This might happen because knowledge workers do not believe in the current process, or because people are unwilling, or maybe because the process is not really sponsored by management and the urgency of some requests trample on the process.

Agile is a solution for this situation; however, we need to mold the process so it can answer the company's needs, bring value and always be respected no matter how urgent an issue is. The process can never be an impediment and it needs to be respected and followed at all times. Naturally, it is crucial that management supports the process and that it is understood that even in difficult times following the process will always be more rewarding than skipping it.

3.4.1 Contact with the client

It is crucial for the knowledge workers that have not experienced contact with the client, or that experienced it a long time ago to go and see the product in use by the client/user.

It would be an eye opener for developers to see the user using the product, understand its difficulties and wishes. This is an initiative that depends not only on the company availability for it as well of the developers will to learn more.

Learn more is not something that all knowledge workers understand as a need, and we need to work more on making them see that we all have something to learn and to grow.

3.4.2 Transversal Resources

From my interviews I understood that many knowledge workers were not happy when they were borrowed to other teams, even when within the same BU. It should be clear that we all represent a team and that if needed knowledge workers should embrace helping other teams, for a period of time, as a new challenge and as an opportunity to grow and understand that a lot can be learned and taught with this experience.

Knowledge should be shared between teams and nothing better than working together to enable this to happen. Of course that these temporary assignments should be properly communicated to teams and members and taken into account when planning.

Rachel Davies, author of the book "Agile Coaching" (Davies and Sedley 2009), has even shared in the conference Agile Portugal 2014 how in the company she coaches, Unruly, they "share" knowledge workers with other companies, so both companies can learn from each other and grow. It is something like an exchange program with other company. One developer of Unruly works at another company for a week and a developer from that company works at Unruly for a week, not in the same week.

3.4.3 Cross-functional Resources

In my interviews I saw that many knowledge workers were not motivated because they were working on the same technology for many many years and they wanted to grow.

We have teams with limited resources, one member for each layer per example. When knowledge workers are so specialized to a layer, and they do not know anything about the other layer, if one of the members gets sick or goes on vacation, no one will be able to resolve issues from that layer in that team. This is a critical factor that can deeply influence client's satisfaction, since urgent matters might not get fixed because no one knew how to fix them.

This is not only true inside the team, as it is between teams; there are teams that are specific to a part of the product. If by any reason all team members are unavailable, there will not be anyone that can work on that part of the product.

The will to learn another technology is an attitude that should come from every knowledge worker, but that should be deeply encouraged, supported and enabled by superiors, team leaders and SPMs.

Within the organization, it is very common that the team leader assigns the issues to a specific team member according to his experience and layer, the reason for this to happen is because knowledge workers are too specialized and it is not understood that a bit of technology transversality can go a long way. The importance of cross-functional resources is clear in many methodologies, Scrum, Kanban, TSP. (Teixeira 2013) (Humphrey and Over 2010)

It is always preferable that the developer chooses the issue he wants to work on and make that commitment, it is a lot more motivational. It goes without saying that this has to happen within the priorities stated by his superiors.

3.4.4 Evaluation

Evaluation is a company necessity and every rank feels the need to evaluate or be evaluated and to have feedback on his work.

In terms of leadership it is crucial to give feedback on the performance of co-workers. This is not usually done due to the lack of an evaluation process and of leadership skills.

This happens essentially because many co-workers have no idea of how their superior(s) see them. Are they all so average that there is nothing good or bad to say about them? Is it that their superior does not understand the importance of giving feedback?

"(...) if you want superior work, you must recognize it and reward it. A key part of every manager job is to be a fan and to recognize, applaud, and reward superior work." (Humphrey and Over 2010)

Individual evaluation should only occur inside the team and never leave that circle; it is important that the team works as one and not as separate pieces and this should be the same for every team. Of course that in critical situations the performance of members needs to be passed to higher management so decisions can be taken.

Evaluation should be a mean for improvement and not an excuse to point fingers at employees. Of course that it should help the management when taking decisions, but it is important to understand that evaluation should always be focused on improvement and not depreciation (constructive and not destructive criticism). Evaluation can never be done using metrics gathered by the evaluees, otherwise they can be tampered with.

Scrum will be very helpful, because the truth is that we need to be evaluated every day, and not only at specific moments. The Daily Scrum meetings are great to evaluate and it is an evaluation that is individualized inside the team as it should be. The team acts as one and the fact that each one of us says what he did yesterday and what he plans to do today and shows it to the rest of the team is evaluation in itself. Moreover, this will be a way for my team members to evaluate me and for me to evaluate them. This is just one of the examples.

3.4.5 Trust

"To build trust, we must start with the assumption that everybody would like to do his job properly." (Humphrey and Over 2010)

Superiors and the management in general need to trust their knowledge workers and, of course, knowledge workers need to show results. This can never be "blind trust", as W. Edwards Deming said, *"In God we trust; all others must bring data."*(Humphrey and Over 2010)

For this issue I have some solutions:

- Scrum daily meetings are very good to make people accountable for their work, people will need to say what they did the day before, and what they plan to do today, and this will make them responsible for their commitments in front of the team. This will work as an accountability measure and as an evaluation measure;
- Sprint Review where the team presents what was done to the Product Owner, thus cementing his trust on the team;
- KPIs (only a few) are currently measured in the BUs, but they are growing and with time they will focus on teams, which will be a very good improvement, and/or will be adjusted to teams accordingly;
- Scrum/Kanban Board – where the work in progress will be clear to everyone.

3.4.6 KPIs

In my opinion, KPIs should never be individual, which would only result in a KPI's manipulation. Individual KPIs would be too objective and inaccurate and people would be afraid of retaliation and constant fear is not the solution to productivity.

At the moment KPIs are divided into Business Units (BUs), and even as so some of the KPI's in place are not real.

In my interviews I have uncovered that in some BUs it is asked to the testers to report found bugs in an unofficial channel, so they are hidden from management. This is a critical failure factor. How can we see improvement, how can we learn if we cheat?

Bugs are a reality that needs to be accepted as it is. Everyone makes mistakes and the idea is to improve on a daily basis, and if there is no real evaluation of the truth, a real improvement can never be actively seen.

KPIs related to bugs need to be accurate so we can see them as real goals that we have the will to achieve, not fake goals that we can cheat.

"If you don't measure quality, you can't manage it. If you don't manage quality, it will not improve." (Humphrey 2007)

KPIs should be seen as team goals and should be achievable but also ambitious; the idea is that when they are achieved they give the team sense of fulfilment and accomplishment and not simply represent a number.

Team KPIs should also be a form of teambuilding, the team should see the KPIs as an ambitious goal that they want to achieve and that they want to present to the other teams as an accomplishment.

KPIs should cover 3 main points at the same time: quality, predictability, and productivity. If they do not foresee these areas they will not be trustworthy. Productivity will be the hardest one to evaluate.

We need to start small and do a lessons learned to fully understand the importance and accuracy of KPIs. Constant improvement is the key to every agile methodology.

3.4.7 Motivation

Evaluation can work as a very strong motivational factor, especially for the average and good performance employees. Feeling their commitment and success evaluated and recognized will be very motivating.

Furthermore, evaluation can greatly help poor performance employees, many of which may not have an idea of their poor performance, due to their nature, and therefore they will see this as an opportunity to improve and grow. Other poor employees, that actually have an idea of their performance but just do not care or do not want to improve, will see this as a threat that will have one of two outcomes: they will try to improve or they will keep as they are and give their superiors the opportunity to separate good from bad.

Evaluation is a necessity that must be implemented, but it cannot be completely objective otherwise important aspects like commitment or proactivity will be lost. Nevertheless, evaluation cannot be completely subjective either. It is important that we find a balance between the two sides.

Evaluation needs to be well thought out and implemented in the near future; this will help motivate knowledge workers to do better and will help the management to have a more motivated, creative and satisfied task force.

"Quality work is not done by mistake, and it is not done by unmotivated or disgruntled employees. The developers must care about their work, strive to consistently improve, and be proud of what they produce." (Humphrey and Over 2010)

There are many things that can be motivational to knowledge workers and the key is finding the right "carrot" for each one.

However with the interviews I understood that there would be an increase in the motivation of some knowledge workers, if some measures were taken, like working toward goals and achieving them, working on a different technology, and having feedback.

3.4.8 Summary

- Knowledge workers should have more contact with the user and the client, but especially the user;
- Transversal resources – helping other teams, or changing teams, should always be seen as a mean to learn and teach;
- Cross-functional resources should always be supported and encouraged according to knowledge workers' will and skills, as some workers will not be comfortable working with different layers;
- Evaluation is important and it cannot be totally objective nor totally subjective, it must be a balance between both;
- Individual evaluation should happen only inside the team, because the team members are the ones that best know the performance of other team members, and the team should be evaluated as a whole;
- Knowledge workers need to be trusted but need to present data so the management can trust them;
- KPIs should be seen as ambitious but achievable goals and never be individual;
- Motivation needs to be a constant work in progress, this is crucial when managing knowledge workers.

3.5 Project Management

Knowledge workers, in general, are not happy on how the planning works at the moment at the company, and feel it is very unstable.

Every change in priorities needs to be evaluated and the impact of such change needs to be taken into consideration. It is understandable that priorities need to change due to many factors, but it is important that each of these alterations is clear to all the stakeholders, so a correct and efficient expectations management can be conducted.

Another aspect which affects planning is that in the weekly planning how much work can in actual fact be done it is not taken into account; it is just the simple organization of work. Naturally, if there is no commitment to do a particular task during that week, it will not get done. Organizing issues in a list to see if we can do them during the week, but not committing with it or even considering if there is enough time in the week for those issues to be executed is

highly demotivating and does not bring any advantages. Knowledge workers need to commit to their tasks and feel ownership of their projects in order to be motivated.

It is important that when planning the work for the week we plan things that can actually be done and that in the planning we take into account that we receive new requests every day. Planning should be ambitious, but when planning we should never forget that new requests are received every day and we should plan with those in mind. The definition of goals makes us less disperse-prone and more focused, thus leading to greater success. One solution to this problem will be the idea of Sprints and the Sprint Planning meeting, where the team will discuss with the Product Owner how many issues they can accomplish in said Sprint.

The Scrum board or the Kanban board can bring big visibility to the work in progress and especially to how many of these new requests enter the work plan in a week (for example); with the pilot project we will be able to understand, within each team, how many unaccounted tasks are requested every week and then be able to adjust the planning for each Sprint.

The personal requests from Operations need to be managed appropriately. Some knowledge workers know how to manage their time and know if they can help or not, others just get lost and try to help their "friends" but get lost and do not deliver when expected. This needs to be better managed, but this will greatly depend on the individual knowledge worker.

In order to help the team leader and the SPM understand the gravity of these interruptions, the daily meetings, plus the board, will represent a clear view of what is happening. How this will be done will be detailed in the proposal.

Another aspect that could bring value to the planning is that it is already known that in "Go Lives" the Development team receives many requests from the Operations teams and this should always be taken into account in the planning. Depending on the teams we could have one or two members dedicated to those requests and the others keeping up with what was planned. There cannot be a constant change in what was scheduled for a particular week; not only because it affects planning, but also because it is a demotivating factor that deeply interferes with productivity.

3.5.1 Requirements

The requirements are always changing and this is not always the client's fault. Legal impositions are always changing and necessities of new markets are a constant due to the company's internationalization.

With this in mind, but as we are not there yet, we still need to do everything at the "factory", there are new requirements every day that are specific to a costumer or market and it is about those requirements that I will be talking about next.

A clear communication with the client and fully understanding his requirements is an issue that affects all software houses (there are several comics that represent this). Some actions that can be taken into account to reduce the impact of the requirements change are:

- Involve the client from the start;
- Accept that the client changes his mind;
- Keep the client in the process;
- Do early prototypes;
- Obtain commitment from the client;
- Manage client expectations;
- Always explain, clearly to the client, what can be done or cannot be done and why;
- If necessary negotiate incremental deliveries, if the client wants a calculator, but we can only, in a first delivery, develop an operation he should choose which operation he wants now and later;
- Clear and detailed description and communication of the clients' requirements to the Development team;
- List the requirements in advance;
- Observe the user using the product to understand the difficulties and what can be improved, like measuring the time the user takes to perform a task – usability evaluation.

It is often the case that development is not synchronized with the requirements because the client changed his mind and nothing besides acceptation and focusing on the bullet points described above can be done. Never forgetting that has been estimated that over half of a product's defects are injected in the requirements and design phases. (Hilburn and Towhidnejad 2000)

However many times developers start the development before the analysis or the design has been drafted, in many cases it has not even started yet. This is a planning problem, which naturally means that the current workflow cannot be executed always, and some teams do not have the capacity to deal with so much requests. If we start developing before the analysis is even drafted there is a big probability that it will not be in concordance with the requirement or that avoidable adjustments will need to be made.

To help with this issue there are several measures that can be taken into account and that is the role of the Product Owner from a Scrum point of view and one of the CMMI level 2 practices (Requirements Management).

It is also very important that knowledge workers perform a good analysis of what they are going to implement and for this purpose the Sprint Planning or Grooming sessions can be very useful since many questions and other information can be cleared here.

The impact analysis is a very difficult issue at the company because the product is very complex and it is not easy to execute an impact analysis especially because nobody really knows the real complexity of the product. There are many non-documented important connections that depend immensely on the person's know-how and other items can only be assessed by looking at the code.

One suggestion that could help in this situation is the realization of formal specifications as this would be a great step towards quality.

Naturally, clear documentation is also a crucial factor; there are many implications that are not taken into account because they were not documented, but this was already discussed in the Documentation section.

3.5.2 Priorities Management

Priorities change very frequently at the company and this can have some negative impacts, it can be disappointing to the client and it can be extremely frustrating to the knowledge worker.

Knowledge workers need to know that they have a long term plan and they need to be able to observe the accomplishment of each step. What they feel is that every day they are interrupted and stopped from doing one thing to start another and this is where the Kanban (Teixeira 2013) idea comes into practice; the idea is that the team should have a limited work-in-progress and a new issue could only be part of the work plan after the previous issue was complete.

I was also able to observe that managing priorities is a very difficult task for the team managers, as I see they have difficulties in stating with clarity the priority of some items regarding other items. Looking at a list of 3 priority items, they have trouble in stating a clear list of items, where the one at the middle has less priority than the one at the top and more than the one at the bottom. Regardless of the priority of items, issues should always be organized in a list of priorities, in which for each item, the item above should have a higher priority than the one bellow and vice versa. The table below, Table 1, explains this.

Table 1 - Items priority. D has higher priority than A, A has higher priority than C, and C has higher priority than B.

1	D
2	A
3	C
4	B

3.5.3 Estimations

Deadlines should never be asked to developers, because they are not in control of their own time, or in the priority of the issues assigned to them.

What should be asked from developers is an effort estimation that can be provided in hours, per example. It is very important to understand that saying that something will take a week to do, this means 40h of work, is not the same as saying on a Monday at 8 a.m. that it will

be done on Friday at 5:30 p.m. Knowledge workers cannot be accountable with due dates because it does not depend on them. This needs to be well understood by management.

Much of the daily job of everyone in the Development team does not depend on them. They suffer constant interruptions, the environments crash, their priorities are altered frequently. There is also some miscommunication, some people receive different orders from different persons and this is a communication issue that happens a lot. Again Scrum Daily meetings will help overcome this matter. If in a period of time we can do A, and we are asked to do A, and B, a compromise needs to be found, because something has to "fall".

Agile methodologies are usually very focused on the client. Scrum pretends the involvement of the client through the whole process, with iterative deliveries, so the client's feedback is frequent and quick. Kanban is based on the Lean methodologies, which is focused on creating more value for the client with fewer resources.

In creative work, such as software development, multitasking and task switching is not very efficient, and can deeply affect quality.

With this I am not saying that knowledge workers should have all the time in the world or that they should not feel any pressure. Pressure is important even as a motivational factor, but only with time, measure and reason.

Knowledge workers should document their work and times in Jira properly as this will serve as proof of their work and some of their complaints will not be seen as excuses. This logged worked cannot be used to evaluate the knowledge worker. It should be used to enhance estimations and perform better planning. Jira must be used as the real tool that it is and reporting times in Jira should be seen as a mean for improvement and not as a bureaucratic and useless task. Knowledge workers need to be seen as trustworthy professionals.

Moreover, there are measures that can help reduce the sense of non-trust and this is where the Daily Scrum meeting, the team leader, the coach and the PROBE method (PROxy Based Estimating) come in. "PROBE teaches developers to consider their products as composed of parts and to estimate the size of each part." (Humphrey and Over 2010)

The team leader is someone that should guarantee that the team complies with procedures; he should protect the team from the outside, but needs to be a superior and not a friend. No one should ever be individualized outside the team; the team should work as one and be seen that way to the outside, to other teams, to SPMs, to heads, and to the world.

This is valid not only for a single small team of developers, but also for BUs and the company in general. The outside world should never know what happens inside our work realm, we should always be seen by the outside world as one. This is an important teambuilding factor that should always be present in everyone's mind.

It is crucial that it is understood by everyone that teams need to commit to their work otherwise they will work just for the sake of it. The team needs to feel ownership of the project and they need to feel involved.

Commitment also works as a motivational factor. The team should commit to a delivery if they have data that allows them to do such commitment. They can use data from Jira and compare times from issues similar to the one they are working on and in future use data from previous Sprints to help them estimate time and effort needed. It is important that complete and accurate data is gathered and analyzed; and it can never be used as an evaluation metric otherwise it will not be reliable. This is yet another reason why KPIs can never be individual.

"The key to motivating such teams (knowledge workers teams) is to have them establish a challenging and motivating team goal and then work hard to meet it." (Humphrey and Over 2010)

In the planning and estimations we should have a common metric. It should be clear to everyone that we do not spend 40h of the work week on tasks and this needs to be taken into consideration when planning. Time spent in meetings, finding answers to our questions, switching tasks, interruptions and restarting a task are not taken into account in the planning and this is a failure factor.

"For example, typical TSP teams find that the amount of time they can actually spend on planned project tasks is between 15 and 18 hours out of a standard 40-hour workweek. The balance of their time is spent in meetings, handling e-mail, helping coworkers, talking with managers, taking training courses, or dealing with interruptions and questions.

When they first use the TSP, most teams find that they had been achieving only 10 to 12 task hours a week before they started measuring and tracking their task time; one large organization ran a study and found that its engineering teams were averaging fewer than five task hours per week before they started using the TSP. With the TSP, knowledge workers can quickly improve their task time." (Humphrey and Over 2010)

I am sure that if we start being more accurate with our time measurement, the time dedicated to task hours will probably only represent around 50% of the 40h week or even less. This happens because knowledge workers spend a lot of time in meetings, reading e-mails, helping coworkers, answering requests from managers, dealing with interruptions and questions, and it needs to be understood that much of this is completely necessary in any software house.

It is important that meetings are reduced to a minimum and in this case the Scrum meetings will be very helpful since they are pre-schedule time-boxed events, which should occur on time. Nonetheless, complete and easy to find documentation is also another thing that would help reduce some of the constant interruptions and requests that teams receive. If we have proper documentation a person that explains the same thing to 3, 4, or 5 different persons will only "spend" the time of writing the explanation once and then redirect people to the answer. Hence, in the long term everyone will get accustomed to looking for the answer and not simply asking.

Another point that I would like to focus on is the macro estimations; macro estimations are extremely inaccurate, they are done without analysis, without small task division and without the real knowledge of implications. It is very important to understand that when these

estimations are made they represent a full commitment to the task and do not cater for the constant drop-ins that need to be handled on a daily basis, and can be completely wrong since much of the information needed to estimate is not available yet. This needs to be improved and doing grooming sessions, experience, and metrics such as Sprint Velocity can help improve these estimations. However, there will not be an instant and immediate improvement and all this work will need time to mature, but if we give up before we really try we will never succeed.

3.5.4 Task-switching

For knowledge workers, especially for software developers, task-switching can be highly inefficient. In software development the developer creates a lot in his head, especially when debugging, he creates a mental model of implications and connections. And when interrupted by another time consuming task the time to once again get to that same state is immense and this is not taken into account in the planning. Not to mention that it is very stressful to lose focus when you are so close to the solution.

The complexity of the task influences the time to refocus and to get to the level of concentration before the interruption. Software development is a complex task, especially when developing a product that is a few years old, that many hands and heads have touched it.

Interruptions and distractions are a reality and are one of the hazards of open spaces, but this type of short interruption has a small impact on the time to refocus and software development needs the communication that comes from open spaces. (Czerwinski, Horvitz, and Wilhite 2004)

What we can do to prevent that too much time is not lost on refocusing and getting to the desired level of concentration again is to deeply reduce the task-switching which comes hand in hand with the change in priorities.

Good documentation will also reduce the amount of time consumed by interruptions, especially regarding interruptions created by the Operations and Support teams.

Task-switching, in software development, can deeply affect product quality, since when interrupted a developer might lose the path he was taking, and might never find it again after the interruption.

3.5.5 Summary

- Dropouts[3] and drop-ins[4] should be clearly represented; for the team these can be seen on the board, but for the Operations team this information needs to be documented in some other form;
- Analysis should always have started before the development starts, this can happen in a Grooming or Kickoff meeting;
- More detailed documentation would help the impact analysis and the development process;
- Estimations must be done in effort, like in hours for example, not as deadlines;
- Knowledge workers are not in control of their time or the priority of their tasks so they cannot commit to deadlines, they can only commit to effort;
- Interruptions, meetings and so on need to be taken into account when planning;
- Critical entries need to be taken into account when planning and time needs to be reserved for them;
- Priorities cannot change so often, not only will this affect the client, but it will also deeply affect knowledge workers' job and the quality of their work.

[3] Îtems that were planned but by some reason were left behing, which might or not be the result of drop-ins.
[4] Items that were not in plan but that by some reason were added to it, which might or not result in dropouts. Both affect the teams and the planning.

3.6 Quality Management

Quality should be the number one focus: poor-quality work is enormously expensive.

This should be the main focus of the company, not only because we are working with critical software, but also because it is a question of marketing and standard image. However, when I asked my interviewees, “Do you feel more pressure with deadlines than with quality?” the majority answered yes. Some used some “funny” expressions like “hell yeah” and this represents more than 77% of the answers. Furthermore, I got some “not in my team, but I know that this is the reality for other teams” (19%) and three “No” answers in more than 70 interviews, which represents 4%, the chart can be observed in Figure 3. This clearly represents that even though quality is a priority for the company, Development teams feel that deadlines are more critical.

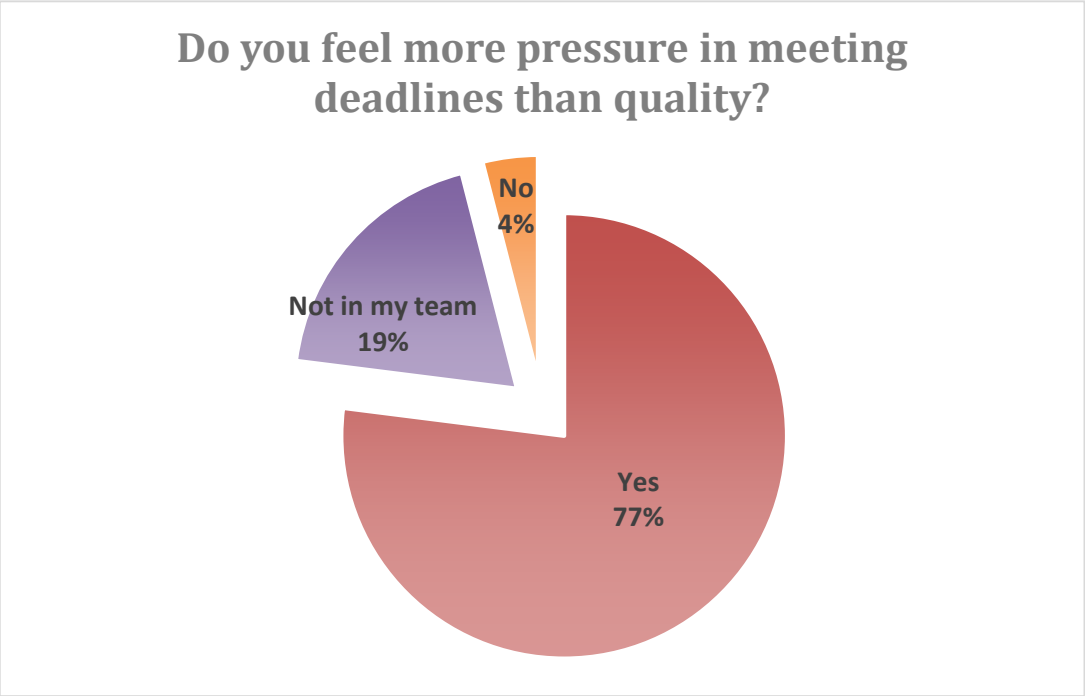

Figure 3 - Percentage of answers to the question "Do you feel more pressure in meeting deadlines than in quality?"

There is no such thing as a defect free product, this is a very strong reality in virtually every software company. This is a sad, but truly reality, however we can work and improve to the point where the amount of defects is residual or even inexistent. Watts Humphrey describes examples of such in his book “Leadership, Teamwork and Trust: Building a Competitive Software Capability”, where we can see results of companies where the Team Software Process was applied and they achieve virtually defect free software. (Humphrey and Over 2010)

TSP has extremely good results relating predictability, productivity, and quality, however is a process difficult to implement since it represents an enormous commitment from the

company, and a very big investment. It is described in many papers and books the quality of the results it produces and in summary the awesomeness of TSP, but it is difficult to make this investment to pass as an investment with big return for the companies' management. (Koch 2005) (Humphrey and Over 2010) (Ferguson et al. 1997) (Grojean 2005) As so there are some actions that can be done in a first stage with less investment.

To help increase the product's quality there are many good practices that can and should be taken that will be described in the following paragraphs.

In a first meeting that we can call the Sprint Planning or the TSP Launch, the items should be thoroughly discussed, with the presence of the Product Owner or similar, analysts, developers, designers, and testers, where everyone can discuss the items in detail and better evaluate them. Many problems that would arise later will be discussed here, thus preventing some defects.

This will also help in the item estimation, think before acting philosophy. In this meeting there should also be a division into small tasks, Work Breakdown Structure (WBS). This is a great way of exploring the item in detail, and revealing otherwise hidden implications and dependencies.

These are two extremely important points that are not done in the majority of the teams. A task is assigned to a developer and some discuss and detail their work, but the majority does not. Many are so stressed to deliver, that they get hands on the item and do not think it through.

Quality should always be the top priority, starting with the developer passing through the middle management and ending the higher management.

Many developers do not review their work, stating they have no time. And it happens that a developer does not test his work enough before committing it to the test. This kind of situations should never happen, or in a worst case scenario, be only the exception to the rule instead of the rule.

A personal review of our work, even before testing it, can represent a significant reduction in the defects detection.

Part of the company's motto is "Double-check", this meaning that you should check your work, and then it should be check by another person. This is what TSP calls the team inspection and that has a serious impact on defects detection.

However this is the motto of the company, only a residual part of the development teams does it. This happens in my opinion because it was not passed to the development teams the importance of double checking/team inspection, they did not receive support or pressure from their superiors to do it.

This double-check should be performed by a team member and in the development stage, meaning before the item is transferred to the Quality Control (QC) stage.

Another practice that could be implemented is the creation of a Lessons Learned sessions, were every member of every development team should feel comfortable to tell and share his/hers good and bad ideas. This should be always seen as a mean for improvement and never

as a form of assigning blame or pointing fingers. In a first stage the Sprint Retrospective can work as a form to make the knowledge workers feel comfortable with the idea of Lessons Learned.

"The key is to recognize that all people make mistakes and that our processes must be designed to minimize those mistakes and to find and fix these mistakes before they can cause serious harm. Until organizations learn to correct mistakes without affixing blame, the fundamental trust problem will continue, and workers will attempt to hide problems instead of helping to identify and fix them." (Humphrey and Over 2010)

3.6.1 Double check

When I discovered that part of the company's motto was double-check I was astonished, I could not believe that such a great practice was put into action in a real software company. Soon I discovered that unfortunately this is not a transversal practice.

Software developers have no time to double-check because of the constant requests and also because they do not give double checking the importance it deserves. This may not necessarily be their fault because very often their superiors do not encourage or even sponsor double-checking. Having said this, it is important to refer that there are teams where double-checking/peer review actually happens.

Peer review practices, such as double-checking, are an enormous quality factor that can exponentially reduce the quantity of defects. However, no time or opportunity is given for it to happen by superiors, thus knowledge workers do not understand the importance it has and do not feel the needed support in order to follow it through.

This is an action that could increase quality immensely but that will not be easy to implement as it would represent a significant change in the state of mind of knowledge workers as well as the management.

The Cost-of-Quality[5] is something that is not well understood or at least is not seen by the importance it has. The idea of Team Software Process (TSP), for example, is to produce such high-quality work that the testing time is reduced.

It is vastly more expensive to find defects in testing or even at the client than it is during the development phase and this can be exponentially reduced with peer reviews such as double-checking.

[5] Cost-of-Quality represents the cost of assuring quality, which increases with redone work. The Cost-of-Quality of detecting a defect when the product is at the client is exponentially higher than finding it in the development stage. More precisely it is the sum of failure, appraisal and prevention costs.

3.6.2 Cost-of-Quality (COQ)

Some KPIs will help in the analysis of COQ and this is a very important aspect not only in the knowledge workers' point of view but also for the management and company's profit.

"(...) the time and money wasted by poor-quality software work often exceeds 50% of total project costs. (...) Because 70% to 80% of the costs of final testing and customer support are typically due to poor product quality, and because these costs often exceed the total cost of product development, showing the developers how to improve product quality can save large amounts of money"

According to the COQ measure, quality costs have four components: (Humphrey and Over 2010)

1. Internal failure costs – the costs of fixing defects during product development;
2. External failure costs – the costs of fixing defects after product delivery to customers;
3. Appraisal costs – the costs of reviewing or inspecting products to find and fix defects before testing;
4. Prevention costs – the costs of identifying the causes of defects and changing practices or procedures to eliminate those causes.

Testing is of course very important but knowledge workers must feel personally responsible for the quality of the products they produce. Quality needs to be incited to knowledge workers; double check/peer review will certainly represent an increase in quality and a reduction in COQ.

"For organizations that do a substantial amount of software work, quality costs are typically 40% to 60% or more of total development costs. And this doesn't even include the large and widely variable quality costs for customer support and service. For a software-intensive business, a large part of these support costs are due to the poor quality of the software development work." (Humphrey and Over 2010)

With the time frame of the pilot project and with the KPI's in place we will probably not be able to see a reduction in COQ, since this type of effect will only be seen in the medium to long term.

3.6.3 Training

Training is a strong motivational factor. The company does functional, product, and health and safety at work training but the Development teams feel the need for technical training. According to the majority of the Development team members, technical training is a definite gap.

This training can be internal, prepared by the Architecture team or even some practical exercises prepared by the Development teams, so the adaptation of a new member becomes more fluid.

Of course external training, like Oracle certifications, for example, would be another motivational factor, but this represents an expense and as so needs to be evaluated by management.

Sessions of lessons learned and good practices, for example, is a simple step towards better product quality.

I would like to focus on the fact that internal training between teams and lessons learned sessions would be something that would bring much value to knowledge workers, developers, and analysts. In the long term this would represent an increase in productivity and especially in quality. The same mistakes would not happen again and again, and functionalities already developed or partially developed by other teams would not be redone and would be availed.

3.6.4 Summary

- Quality must be the main focus;
- Everyone makes mistakes, we need to minimize the mistakes without affixing blame, otherwise every metric will be inaccurate and we will not be able to improve;
- Bug reporting should be accurate, we need to be able to evaluate improvements and failures;
- Double checking is a great idea that should be put into practice in all teams, as it can deeply reduce de Cost-of-Quality;
- Cost-Of-Quality needs to be evaluated and measures to reduce it taken into account, such as Sprint Planning meetings, Sprint Review, Sprint Retrospective and personal and peer review;
- Environments should be stable and have the correct contents available in order to perform proper testing;
- Training should occur often, especially internal training in good practices and lessons learned;
- We can learn a lot from mistakes.

3.7 Summary and Conclusions

The smallest summary we can say is that there is much that can be done. We will not try to resolve every problem, but we will start small and work our way up to a better work life. After all *"nothing is particularly hard if you divide it into small jobs"* (Henry Ford).

The main conclusions of each chapter are seen in its summary. However, I would like to bring the focus to some points for high-quality work.

The solution in order to enhance and have more and better quality is not to hire more testers or have more time for tests (Humphrey and Over 2010). Of course tests are very important, as testing time and beta testers are great to find bugs. But the key to better quality is developing a high-quality product and this is why I would like to focus on the following points:

- Good and open general communication;
- Good analysis, in particular, impact analysis;
- Dividing the work into small tasks (WBS);
- Reviews (personal and peer);
- Lessons learned without pointing fingers;
- Planning so that there is not a constant change of priorities and we do not lose focus on tasks;
- Setting achievable but ambitious goals so that knowledge workers are motivated;
- Having a process which will help attain high-quality work and is respected even when everything is "on fire";
- No one should ever trample on the process, not even management;
- Process must be adjusted to the company's reality and be focused on the product's quality, but respected and followed always.

The greatest conclusion that I found with my interviews was that everyone recognized that they are not doing their job the best way they could, unfortunately they are still very far from it, but many are willing to change and do something in order to effectively change.

Of course I also felt much resistance to change but the majority was not regarding my project or my ideas, it was regarding the sponsorship of my project. Knowledge workers do not believe that change will happen because they feel that something more important will always appear.

I will do the best I can to prove that things can change and that we can get to a better place, produce a high-quality product and have the client and knowledge workers more pleased and satisfied. But this, unfortunately, will not depend only on me, first I will need the management's

support and sponsorship, the teams' commitment and everyone's willingness to change and do a better job and have a better work experience all round.

A critical success factor to this project is the management's sponsorship not only because it is referred in the books (Berkun 2008) (Humphrey and Over 2010) but also because it was what knowledge workers pointed out as one of the difficulties towards achieving a better development process, better quality, and a better work environment.

The principal leadership elements that should be provided, so that this change program enables us to excel in this exciting but challenging future, are: Goals, Support, Motivation, and Standards of excellence and Execution. (Humphrey & Over, 2010)

This diagnosis was validated by members of the company that sponsor the project taking into account their expertise in the company's situation and experience.

Chapter 4

Proposal

Although in some cultures might work just to tell the knowledge workers *"from now on you'll starting doing things like this"*, and it works, that is not usually truth. Knowledge workers need a different type of management and need constant motivation.

This topic represents the proposal customized to the diagnosis, and company's reality, understanding that we can't blindly apply processes. We need to deeply understand the company's culture and DNA and adjust the process to it.

Agile is about continuous improvement, and we can only achieve excellence if we keep getting better and better and identifying areas of improvement. As so this proposal will be iterative; for now we will have a starting point, but only during and after the pilot we will really understand the impact it had. Throughout the pilot process I'll be adjusting the proposal, following the agile principles, and in the end of the pilot we will see more things that we can improve, it is a never ending process.

The focus of this proposal will be quality, however we cannot achieve high-quality product with demotivated workers, so motivation will be something that this proposal will try to bring. (Humphrey and Over 2010)

In each topic I will do a brief description of the practice and explain why we should implement it.

This chapter starts describing a brief scope of the proposal which is followed by Scrum practices, the Kanban Board, Team Software Process (TSP) practices and other important good practices, which relate to the 3 methodologies.

CMMI was took into account in the diagnosis, and it influenced some of the choices made for the proposal.

This proposal is the second most important output of this dissertation, because it is customized, and focused on the diagnosis. The aim is that the proposal covers some of the main

areas of interest, focusing on the more critical known problems without damaging the things that make this company a success.

4.1 Introduction

It's not a good idea to blindly apply a methodology to a company; every project is different and this proposal is focused for the development of the company's product in the reality of the more critical teams. This proposal was developed focusing on the reality of those teams, however it might work in other teams, but tweaks might need to be done.

This is the initial proposal that was applied in the pilot teams. Agile DNA is continuous improvement so from sprint to sprint during the pilot, and in the future if the proposal is expanded to other teams, we might need to adjust a point or two. There are no secret recipes that are full proof and deliver awesome results.

This proposal will have as the main focus Scrum, although it will not be Scrum. It will be a merge of Scrum with Kanban, and getting some other good practices from Team Software Process to bring the focus on quality.

CMMI was taken into account in the choices made in Scrum, however it is important to understand that they work at different levels. This proposal is more focused on the micro management and CMMI is more on the macro management. Also CMMI gives guidelines but not the how, and the how we get from this proposal. (Garzas and Paulk 2013)

Since the focus of this proposal is agile principles, it is never too much to remember what referred as valuable in the Agile Manifesto is (Beck et al. 2001):

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- *Individuals and interactions over processes and tools;*
- *Working software over comprehensive documentation;*
- *Customer collaboration over contract negotiation;*
- *Responding to change over following a plan;*

That is, while there is value in the items on the right, we value the items on the left more."

With this being said there a few misconceptions that I would like to state:

- Agile does not mean lose focus;
- Agile does not mean work without method;
- Agile does not mean defy common sense;
- Agile does not mean everything is on the flow;
- Agile does not mean chaos!

Before the implementation of the pilot, training in the proposal was given to the participating teams. In the eventuality of the acceptation of this proposal, all the teams will received the same/upgraded training.

4.1.1 Scope

Before the diagnosis was conducted, several meetings were conducted between me and the dissertation proponent, which made me see some possible problems, and thus guiding the interviews script and limiting the scope of the proposal.

At that point the following points were already known:

- The number of defects is not satisfactory;
- Estimates are inaccurate;
- Priorities are always changing;
- When they tried Scrum, "Sprints" did not have a fixed time, it was assumed inside the company that Sprints corresponded to versions, so they vary in time-frame;
- Team members do not participate in the estimates;
- Time for analysis and tests are not included in the estimates;
- Developers did not feel ownership in the projects.

With these issues in mind, and to restrain the scope of the proposal, three referential where taken into account: CMMI, Scrum and TSP.

Why choose these three referential?

CMMI was not actually chosen, it is a requirement, and the company wants to increase its maturity by the end of 2014 and achieve the CMMI level 2 maturity.

Scrum was the initial idea of the proponent and since the company has the need for readapting itself to needs of the market and stakeholders. Considering the reports in several studies (Potter and Sakry 2011) (Hansen and Baggesen 2009) (Garzas and Paulk 2013) about the "complementarity" of Scrum and CMMI, Scrum was also chosen to be part of the mix.

The third referential, TSP, was chosen due to the need to focus on quality, although it was also reported in the study "TSP can be the building blocks for CMMI"(Koch 2005) that TSP can speed up the CMMI maturity level achievement.

For these reasons and to restrain the scope of the proposal, and of the dissertation those where the chosen referentials.

However in the analysis of the diagnosis, it was clear that Kanban was also needed in this scope, and as so it was added to it.

4.1.2 Proposal validation

When the proposal was complete it was validated by dissertation proponent, Rui Borges and by Software Engineer expert João Carlos Pascoal Faria, PhD – Teacher and researcher at Engineering Faculty of University of Porto, Certified PSP Developer, Authorized PSP Instructor, and Trained TSP Coach.

4.2 Software Process Coach or Agile Coach

The idea of having a coach was, in the beginning, inspired by the TSP coach, however during the project this role was assimilated as an Agile Coach.

This is the part that I'll be representing in pilot program. My part is a bit similar with the Scrum Master, but in a more broadly manner.

My main objective is to keep the teams focused on following the proposed process, not only the Scrum, or Kanban, but all that is proposed. I will be constantly evaluating the impact of the practices adopted and participating in all Scrum ceremonies, especially in the Sprint Retrospective meetings to see what teams have to point out as improvements, accomplishments or problems.

This role is also responsible for keeping teams, and if possible management, motivated with the development process. Remove any obstacles that may arise during the process, represent a support for the teams, and be a bridge between the teams and management.

It's also part of the coach's job to help team leaders/Scrum Masters to do theirs, help them motivate the teams, maintaining appropriate standards, and building team cohesion, otherwise performance will suffer. (Humphrey e Over 2010)

And of course, the constant improvement and inspect and adapt philosophy so present in Agile DNA, will be on the coach's shoulders. Analyzing how the teams are adapting and using the proposal, sharing the knowledge and practices between teams and also researching how to improve and overcome problems.

And as someone stated in one of the interviews, be the policeman that will make the process, and establish good practices, to be followed at all times.

4.3 Scrum

The choice of Scrum was made because change is the everyday reality of the company, and change needs to be embraced and not ran from. We can't hide from change, we need to expect it, and be prepared to embrace it and work with it, and this is why not only we are partially adopting Scrum as we will also adopt the Kanban board to help with the constant drop-ins in the planning that are an everyday reality.

Wrong assumptions:

- Sprints will not be black-boxes where no new things go;
- Functionalities will be "delivered" to QC as they get done an not only at the end of the Sprint;
- Sprints will only be planned for a percentage of the duration, the other percentage will be for unplanned tasks;
- Quality can never be negotiated- we can negotiate scope, importance, and estimates, but never quality. (Kniberg 2007);
- Since it will be impossible to adjust the Sprints to the releases, there will be items that the team will commit to deliver in specific dates.

The following topic will describe how we will implement Scrum in the company and why.

It is also very important to refer that one of the core values of Scrum is transparency. (Kniberg 2007)

4.3.1 Roles

We will implement the 3 roles of Scrum, Product Owner, Scrum Master, and the Development Team, which will be referred in this document as Team.

Product Owner

In the company's reality this role should be done by SPMs coordinated with their superiors. This role is a very important so clients' goals and wishes are always in focus. These wishes and goals might not only represent the wishes and goals of the real client, but also goals and wishes that the company's has, as a mean for product improvement and competition differentiation.

Value to the users or customers – this is on the pillar of Agile – focus on what the customer really wants. User Stories must bring value to the customer and he should easily acknowledge it. Customer value is very useful information when it comes to User Story prioritization, and the Product Owner is responsible for it. (Teixeira 2013)

Scrum Master

This is a role that in the company's reality can be played by team leaders, however in the pilot project it will be supported by the Software Process Coach, which I'll be representing.

Team

In a first stage we will not make any alterations to the existing teams and as so testers and analysts will not be part of the Team, which will be something that will be analyzed in the future.

However they are not part of the Team, they should be present at least in the Sprint Planning Meeting. The presence of functional analysts is important, since they are the ones that best know the product, and that analyze the clients/company's requests. Testers should be present to better understand what is being planned and to better prepare to test it, allowing them to start with the creation of appropriated test cases.

The fact that the testers do not belong to the team should not affect the testing. The Team should always thoroughly test their developments, before and after integration.

4.3.2 Events

One of the big advantages of implementing Scrum at the company is its ceremonies. As I described in the diagnosis, communication is one of the problems that affects the company. Scrum ceremonies, are time-boxed events, planned in advance that should happen according to schedule.

Scrum ceremonies, will help to improve communication, between team members and management, and hopefully between teams. They will also help reduce unscheduled meetings, nonproductive meetings and the time wasted in those. To regulate the Scrum ceremonies, the Scrum Master and the coach roles are fundamental.

Sprint Planning

It is important that when doing the Sprint planning we take into account the company's reality of new drop-ins every day, as so the planned items cannot completely absorb the Sprint's capacity. From Sprint to Sprint we will understand how much is the percentage of time that we need to save for drop-ins. This is something that will not be perfect in the first run, or in the second, is something that needs to be adjusted to the teams and phases of the project, depending if we have "go lives" or not, per example. This is one of the things that will be in constant improvement and adjustment. In the Sprint Retrospective meeting we will need to evaluate whether the percentage met the drop-ins needs or not.

To help with the choice of Product Backlog items and to help understand actually how much time it will take to do the tasks please take into account what is describe in the topic Work Breakdown Structure.

Daily Scrum (Daily Standups)

To the 3 questions defined in the Scrum Guide (Schwaber and Sutherland 2013) I consider that in the company's reality it is important to add a 4th question: What did I do yesterday that was not planned?

This extra question might get answered in the question "Do I see any impediment that prevents me or the Team from meeting the Sprint Goal?", but it is important that this point is focused, because one of the difficulties that middle management had is that many times the

teams would only communicate tha they were late in the delivery day, so it is crucial that the teams understand, as soon as possible that they are going to be late, and that they understand how important it is to promptly communicate that to their superior.

In the company's reality this is one of the practices that in my opinion will bring more value. Because it will serve as a way to keep the teams on track, assuring that the defined goals are achieved, to take proper action regarding the drop-ins, and serve as an evaluation, so needed in the teams.

Today we don't assume commitments, and the constant new requests are not documented or even communicated which brings delays that are not accounted for and provoke slippages.

Sprint Review (Demo)

In the company's reality, as well as in many companies, it will happen that some features do not fit in one Sprint, however, as Henry Ford said *"Nothing is particularly hard if you divide it into small jobs"*, and we can to this division and then accept each small part as "done".

It might happen that we need a different definition of "done" for some items, since the same definition will not be applied to all.

Sprint Retrospective

In the pilot the retrospectives we will focus many aspects, especially regarding the following topics:

- Difficulties that the team has with the implementation;
- Comparing estimates with logged time in post-its and with actual work logs from Jira;
- Discussing the Kanban board;
- Discussing/Analyzing what the Team feels about the proposal and how it is fitting/satisfying their needs;
- Discussing issues that affected the team performance;
- Analyzing Kanban wastes;(Ikonen et al. 2010)
    - Waste A: Partially done work;
    - Waste B: Extra processes;
    - Waste C: Extra features;
    - Waste D: Task switching;
    - Waste E: Waiting;
    - Waste F: Motion;
    - Waste G: Defects.

Grooming

Grooming is very important to expedite Sprint Planning meetings, as so teams should reserve 5-10% of the Sprint's capacity for it.

Since we will adopt two-week Sprints, most of this grooming will probably happen in the Sprint Planning meeting, but it is important that at least, the Product Owner shares in advance the Product Backlog so the teams can better prepare for the Sprint Planning meeting.

Sprint

In the company's reality we cannot have Sprints with a big time-box, because it simply would not work with the constant alterations and short deadlines that appear. Sprints cannot be as short as one week because it would result in low productivity due to the necessary meetings. So the best approach for the company is two-weeks Sprints. Big enough to justify the meetings and small enough to manage the constant changes to the planning.

This will be the time-frame that we will apply to the pilot teams since it is the more appropriated to them, however this time-box may not be the same for all the company's teams. If in the future we expand this proposal to other teams, we will need to verify that two-weeks Sprints is also the way to go to those new teams. However, from what I was able to understand from my interviews, two-week Sprints is the best option for all teams.

There is however, a golden rule for Sprint lengths, is that they should be as consistent as possible, this meaning that unless big events prevent it, like 90% of the team is on vacation on the next Sprint Planning, per example, Sprints should always have the same duration.

4.3.3 Artefacts

Product Backlog

The company works with many different clients and markets that have different needs or customizations. As so a very big alteration and an important one that I suggested, was that the Product Owners had one and only one Product Backlog, maintaining it in constant prioritization. (Kniberg 2007) Now there is not a notion of what has higher priority regardless the market or client.

With this practice it would be a lot clearer the impact of any alteration in priorities. We would clearly see the impact that moving something to the top of the list would have.

Of course that this one and only Product Backlog should not only be at the level of the Product Owner, represented by the SPMs as it should be at least in one level above, the department Director.

Since it happens that teams may work in more than one version per Sprint, it's important that at least the SPM's Product Backlog includes all the work for all the versions. So priorities are organized according to the market/client's necessities, and no trampling of priorities occurs.

This is going to be maybe the biggest challenge of the project, oblige that all the priorities are aligned regardless the market/client/version it belongs to. This priority alignment will be a big challenge that will take a lot of effort and commitment from management.

A common Product Backlog will represent a clear view of what are the company's priorities, which client we are all focusing and why. It will bring alignment in the teams and a clear and visible communication of the goal, which is motivational. It may also bring productivity gains, because teams would be more goal oriented and not running to a different front every day. Things will be planed and visible. Not only to the teams but to management, it would be very clear the impact of alterations in the prioritization.

When something gets pushed-in in the backlog, it will be very clear the items that are pushback or even put aside. Now these "dropouts" get lost in the shadows, and many of the issues we have regarding planning is that things fall into the shadows and when we go there getting them they are on fire, and then another item goes into the shadows. It's a vicious cycle that keeps happening time and time again, and a common Product Backlog, even partial, would bring some of the needed visibility.

Sprint Backlog

To the pilot teams I propose that we plan for 50% of the time available, in Sprint 1, and inspect and adapt to the next Sprints. This is a necessity since we will receive new requests every day that need to be answered to and it is crucial that we take these drop-ins into account and with time master the amount of time reserved to them so that what was planned always gets done.

We will replace the common Scrum Task Board by a Kanban Board.

We will also include one item from the Technical Backlog in the Sprint Backlog, which should correspond to 10% of planned time, to reduce the technical debt.

Burndown Chart

Burndown charts will represent only the effort related to the planned items, Sprint Backlog. The unit measure will be man-days, as it corresponds to the estimates unit measure and as it should be well understood for everyone in the team.

4.3.4 Estimates

In the pilot program we will adopt the Planning Poker method referred in the topic **Sprint Planning**, meaning we will be estimating in man-days in which a day should be correlated to a workday for one person.

I also propose that if an item has an estimate superior to 5 days, that item must be divided into subtasks. So we can estimate better and understand all the implications related to it.

All tasks in the Sprint Backlog, including tasks from the Technical Backlog, need to have estimates.

With time, and looking at Sprint velocity, and to times in Jira we will get better and better. This is also a learning process.

4.4 Kanban Board

As was said a physical board has an enormous impact in agile methodologies. This is also what a company's team that in the past experienced Scrum, said. The team stated that the Scrum board was a benefit, and that they loved the post-its.

In the company's reality I do not think that a simple task board will suffice. The company produces a so wide and complete product, present in so many markets and clients, which has, as many other product companies do, constant client requests, firefighting, the necessity of getting hands on unplanned items, e.g., demos for new clients. As so it needs a board, which not only facilitates the resolution of all these problems as it helps the team to deal with so many requests.

Kanban board has some particularities that help in dealing with task-switching, expedite requests that need to be taken care quickly, queues to protect the team of being overwhelmed with partially completed work, and so on.

This is why I propose to adopt the advantages of the "leanniness" of Kanban and merge them with the oriented agility of Scrum.

As an integration with CMMI it is required that the Scrum Masters take daily pictures of the board, always at the end of the Daily Scrum meetings, to act as evidence. (Potter and Sakry 2011)

4.4.1 Visualization of team work

This will be something that will bring many value for the teams, not only for the teams in the pilot program but also to the surrounding teams.

Per example, if testers do not belong to the Scrum Team, this will provide them with a quick view of what needs to be done next, even without them participating in the Scrum meetings. This is one of the aspects that makes this board so crucial in this project.

However this first design of the board may not be perfect, but as other aspects of this project it is always in continuous improvement. The number of columns as the capacity of those might be altered in the learning process.

4.4.2 Limit the work in process (WIP)

This is a key point in Kanban and a necessity in the company's reality.

As was stated in the diagnosis, multitasking and task switching is not efficient. And to pile up tasks without limiting WIP increases the duration of the tasks, hence increasing the lead time, which significantly affects productivity and can affect company's profitability!

As so this an important advantage of Kanban that will be very useful in the company's reality.

In a first stage I recommend that the capacity (WIP) of each column correspond to the number of developers/testers/analysts available for the project. This is something that we will work on to understand what works best, but at the start is what we should do.

4.4.3 Only start new work when an existing work is complete

One of the teams that I've interviewed follows this rule, and they affirm that they have a better work life because of this. And it is clear why, because focusing on one task at a time, results in higher quality, than doing many at the same time.

Multitasking and task-switching was appointed as one of the difficulties that the knowledge workers have, and is expected that this measure helps in that matter.

4.4.4 Kanban Tickets

As tickets represent small tasks, preferably independent, more easily manageable, if they belong to the Sprint Backlog they should differ from other tickets, which can be in color or shape. All Sprint Backlog items should have an estimate.

Drop-ins should be identified also, as in color, shape, or even a different swim lane. In the pilot we opted to divide the board in half, horizontally, half for planned items and half for drop-ins. Drop-ins might not have an estimate associated with it as they have a degree of urgency that makes them be critical tickets.

Since we have two-week sprints, when we get drop-ins they will need to be processed quickly so they will act almost like Expedite items.

In the company's reality I suggest the usage of swim lanes to identify priority, like with a simple scale, high, medium and low, meaning 3 swim lanes.

As the board will also include drop-ins, my advice for the drop-ins is that if they are complex enough to absorb more them a day's work they need to be divided into subtasks, otherwise they can represent only one ticket.

4.4.5 Pull system

As was said in the state-of-art review, this system allows us to visualize bottlenecks and thus understanding which are the steps that need more attention. This system will be an eye opener for bottlenecks and workflow impediments that, at the moment, might not be clear.

4.4.6 Cycle-time and Lead-time

These two measures are very important in Kanban, but might not be completely functional in our reality, in an early stage. We can use cycle-time to help us understand if our estimates are in concordance with the truth, however it will be a lot easier to analyze this with the work logs in Jira.

Lead-time can be very helpful in hotfixes, to determine how much time we will take to deliver it to the client in the future, however we know that tasks can differ a lot from hotfix to hotfix.

4.4.7 Class of service

We will adopt 4 classes of service, Expedite, Fixed Delivery Date, Standard and Intangible.

As Sprints will not correspond to releases, be it version releases or hotfixes releases, it is crucial that if an item has a different deadline than the Sprint we clearly represent it in the ticket, this representing a ticket with the class of service Fixed Delivery Date.

As we will have drop-ins that will usually represent an item that needs quick action, this tickets will, in the majority of cases, correspond to the class of service Expedite.

One of quality measures that we will adopt, to decrease the technical debt, will be the usage of a Technical Backlog, that will have technical items, such as performance improvement, as so this will clearly correspond to the Intangible class of service.

For all the other items, that do not correspond to any of the above we will use the Standard class of service.

4.5 Good practices inspired in TSP

TSP focus many aspects that walk in pair with agile methods, like self-managing teams, kickoff meetings, communication, cross-functional teams. But in my opinion the biggest advantages of TSP, in this company's reality, is quality and the accuracy of estimations.

TSP bases on the complete and accurate acquisition of data to do high-quality work.

We already have in place, at least in theory, some of the good practices reported in the TSP books, however they need some refinement and in some cases to pass from theory to practice.

4.5.1 Data acquisition

Its company police to register time logs associated with tasks in Jira, however, some people do not do it in an accurate way. Because there are no rules defined to them of how to report that data, we need to establish rules. This is a practice that should continue, because it will help management predict completion times, and will also help knowledge workers to do better estimates.

Since one of the complaints of many knowledge workers was that their work machines were slow and outdated it is important that all software and hardware problems are reported in the specific task that exists in Jira for it. This action will help management understand which are the knowledge workers that need a hardware improvement and also help management make the necessary calculations to understand the impact of an investment on new machines.

All time spent on an issue, be it discussing it, developing it, testing it, should be logged on that particular issue with an appropriated subtask or comment. This will require discipline, and will take time for the knowledge workers to get used to such detailed data, instead of logging some of those times in more generic tasks such as support to teams.

It is extremely important however that this data is not used to evaluate the knowledge worker or to be used in a depreciative way, otherwise it will be inaccurate and dubious. (Humphrey and Over 2010)

4.5.2 Quality must be the Top Priority

"An effective quality program can save a lot of money; if the quality program doesn't save money, it is not being managed properly. A third point, which is part of every topic we discuss in this book, concerns measurement. If you don't measure something, you cannot manage it, and if you don't manage something, it will not improve. Because few people measure quality, most quality improvement programs are just motivational talk and produce no meaningful benefits. This is why any serious quality program must start with measurement." (Humphrey and Over 2010)

Not doing something right the first time is extremely expensive. The more time a defect stays in the product the more expensive it becomes. Finding a defect in the development phase is a lot cheaper than finding it in the test phase. And the gap in cost between the analysis phase and live in the client is stunningly wider, and in a mission-critical software, not only represents money, as it can be indirectly life-threatening. So quality should, undoubtedly, be the number one focus.

"The reason is that the cost of finding and fixing a defect in a software system increases exponentially the longer the defect is left in the product. This is illustrated by the Xerox quality data shown in Figure 4. These data were gathered by a Xerox TSP team during development of a software system." (Humphrey and Over 2010)

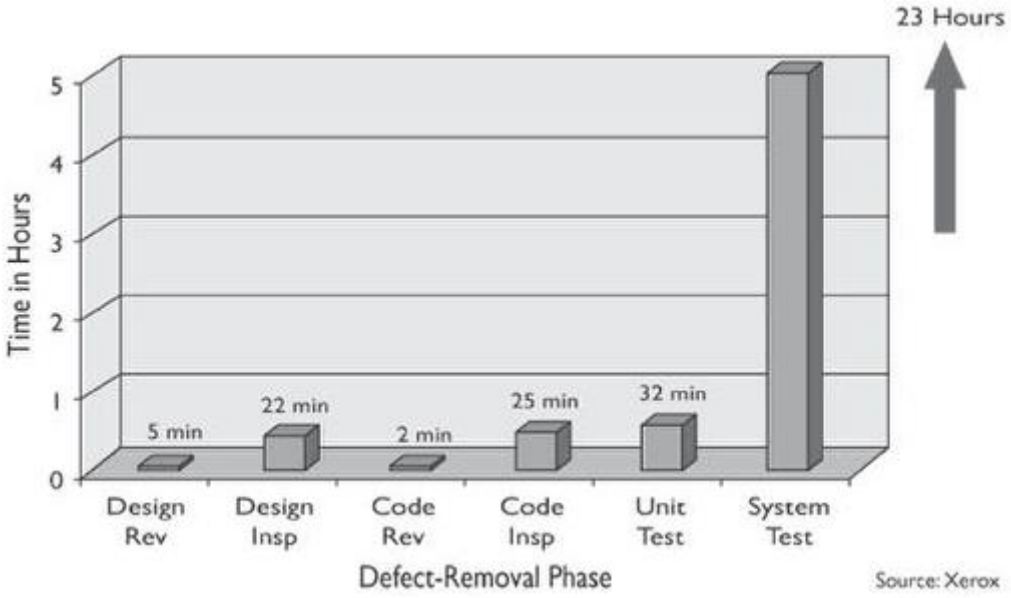

Figure 4 - Xerox removal time per defect (Humphrey and Over 2010)

"By adding these (see Figure 5) review and inspections steps, TSP teams typically find and fix 95% to 100% of all the defects in a product before releasing it to system testing." (Humphrey and Over 2010)

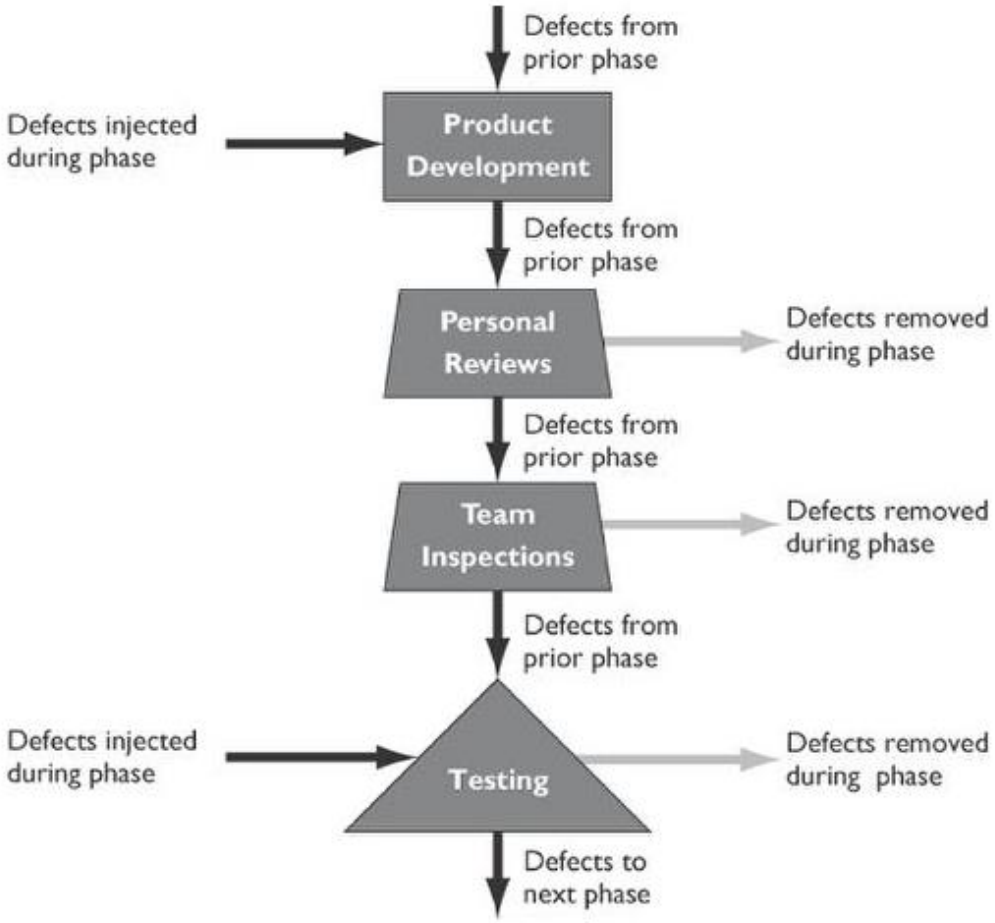

Figure 5 -The TSP quality process (Humphrey and Over 2010)

"To improve product quality, sharply reduce testing time, and cut costs, your software developers must follow a quality strategy like that shown in Figure 3. In each development phase, the developers perform an activity such as requirements development, design, or code and build. During the development part of each phase, they typically inject defects. Following this development step, each developer then personally reviews his or her work products to find and fix as many defects as possible. After completing this personal review, the developer asks a team of other developers to thoroughly inspect the product to find as many more defects as possible. Only after each product has been reviewed and inspected is it considered ready for the next development phase or for release to testing. The economic benefits of this strategy are best illustrated by the experiences of TSP teams. For example, the defect-removal times for a Xerox team were shown in Figure 2. In that diagram, we can see that it takes just a few minutes to find and fix each defect in the personal review steps, and the inspection steps take about half an hour per defect." (Humphrey e Over 2010)

Resuming quality is the most important factor when developing mission-critical software, and guarantying quality should be always the focus of the organization and of its knowledge workers.

4.5.3 Personal Review and Team Inspection

Pair programing is probably the best way to reduce defects production, however this is something very "hard to sell" to management as it is not uncommon that knowledge workers feel uncomfortable with this approach. Pair programing is not only great to produce high-quality work, as it has other advantages, as increasing knowledge workers focus, reduce distractions, share knowledge, improve knowledge workers skills and knowledge. However we know how hard it can be for any management to understand the advantages of pair programing, as what they usually see is two persons doing the job that one person could do.

As so what I suggest is team inspection, which also has great results in reducing the cost-of-quality.

In order to increase the knowledge of the knowledge workers, and to increase product's quality, I propose that, in the very least, all planned items go through two revisions of work:(Humphrey and Over 2010)

- A personal review (walkthroughs, inspections), where the knowledge worker would review his work, and detect defects; (Hilburn and Towhidnejad 2000)
- A team inspection, where another team member would review the work done, usually in parallel with the knowledge worker that developed that item, and detect defects.

Personal Review

It is a good practice, which should be done no matter what. A personal review should be as natural as wearing clothes, and never skipped.

It's extremely important that every knowledge worker does his/hers review of the work done, this should always happen before a team inspection is done. All defects found this way will be quickly fixable, meaning low costs.

Team Inspection

Team inspection/review should be done to guaranty that the product goes with higher quality to QC and not as a way to penalize whomever created the defect.

This will be harder to implement but will represent great results. The fact that someone in the team, that can or not be the team leader, depending of his knowledge, since not all team leaders are cross functional, should do a review on the code.

This is not to be done in a depreciative way, it's just the company motto "double check" really happening in the development teams.

This is a simple code review, not to verify if the variables name are good or not, it's to find defects and defects only, that can be poor performance, or something more basic like a < instead of <=. This should only happen inside the team and should only be used as a mean for improvement, never in a critical or depreciative way. We all make mistakes, we need to find de causes, fix them and learn from them so it will not happen again.

"The key is to recognize that all people make mistakes and that our processes must be designed to minimize those mistakes and to find and fix these mistakes before they can cause serious harm. Until organizations learn to correct mistakes without affixing blame, the fundamental trust problem will continue, and workers will attempt to hide problems instead of helping to identify and fix them." (Humphrey and Over 2010)

The implementation of these two revisions of work should represent a very effective increase in defects detection previous to the test phase, which will represent a significant reduction in Cost-of-Quality. (Humphrey and Over 2010) This is a measure to deliver a high-quality product and to reduce the cost-of-quality and not a measure to give wings to inquisition. This should be done inside the team, and to help everyone improve and learn from mistakes. If we do not have this approach, people will be afraid to detect defects and hide what cannot be hidden and what we can learn from it, and so it does not happen again.

My proposal is that this inspection is done to the code, and if justifiable also do an inspection of what was develop through the product's user interface.

This would also represent an important and motivational factor, because knowledge workers would start to learn new technologies and languages, share good and bad practices.

In non-cross-functional teams this revision might not necessarily be done by one knowledge worker of the same layer. It will be a form of sharing different knowledge of different technologies, and in the long term this will also represent an increase in the team's

autonomy, since it will become more and more cross-functional and stop depending on specific individuals.

4.5.4 PROBE – PROxy Based Estimation

"PROBE teaches developers to consider their products as composed of parts and to estimate the size of each part. The part estimates are then added together to produce a size estimate for the entire product. PROBE also requires developers to measure the actual sizes of these parts after they have been developed. With data from just a few products, developers have enough size data to make accurate estimates for new parts. There are three reasons that PROBE estimates are more accurate than just guessing:

1. *Breaking their products into parts and estimating the size of each part reduces the likelihood that developers will omit key functions.*
2. *Using historical data increases the likelihood that estimates are unbiased. That is, the estimates are just as likely to be high as low. PSP training sharply reduces estimating bias.*
3. *When multiple unbiased estimates are combined, their errors tend to cancel each other, providing a more accurate total.*

The third reason is somewhat helpful for individual estimators, but it can substantially reduce the overall error in team estimates, assuming that all team members use the PROBE method." (Humphrey e Over 2010)

PROBE will not be implemented in the pilot but it is important that we starting walking to its path.

4.6 Other good practices

In this chapter I'll describe some other good practices that should be taken into account and implemented.

4.6.1 Work Breakdown Structure (WBS)

"Nothing is particularly hard if you divide it into small jobs" Henry Ford

This is an important good practice, not only in agile but in everything. That can be done in the Sprint Planning, for the planned tasks, this way everyone contributes making a better and more accurate division into smaller tasks. Nevertheless it should be done in all other tasks.

1. Diving a task into smaller ones helps in the estimation of the task time (PROBE – PROxy Based Estimation), making it more accurate;

2. Helps to clarify the task, this is also a quality measure, the fact that the Team discusses the process to implement an item diving it into smaller tasks will help to better understand the complexity of the item and if there are implications that do not pop on the first look;
3. It reduces the likelihood of omitting key functions;
4. Combination of multiple unbiased estimates provides a more accurate total;
5. It will help to observe progress in the Daily Scrum meetings and in the board, so a sense of accomplishment is felt in the end of every task, motivating the knowledge workers.

The level of detail of the WBS should evolve throughout the Sprints, and according to teams and their nature. However, in order to have more accurate estimates, I advise that for every item that has more than two days estimate, we detail the task more, in the Sprint Planning. Of course it is advisable that the knowledge worker does a more detailed WBS when developing that item.

4.6.2 Teambuilding activities

Teambuilding is something that is very important in teamwork activities and software development is usually a teamwork job.

This is a company that really depends on teamwork, it has a product so complex and complete that depends on each and every individual, so we need to have a great cohesion between team members.

The Scrum meetings bring a bit of a well needed teambuilding, however it might not suffice, so other actions need to be put into place. This kind of activities need to be chosen according to the team, and should be optional otherwise it can have the opposite result. There are some teams that already do this, like weekly team lunch per example.

It's not my job to impose specific teambuilding activities, however I want to state the importance of those. Each team should choose the activity that fits them the best, and management should encourage and support this type of activities.

4.6.3 Technical debt

The company needs to reduce, or at least stagnate, its technical debt. The company has, of course, quality standards and good practices defined by the Architecture team however many times the easy and quick solution is the one used, and this needs to be avoided at all costs, as was stated before not doing something right the first time around is extremely expensive.

Quality needs to be the Developments Team's number one focus all the time and every time.

The teams will probably need more good practices sessions from Architecture, but small things like Lessons Learned sessions and even the discussion of a task and division of it in smaller tasks will help reduce the technical debt.

We need to stop the introduction of short-term and quick-fixes solutions!

"Let's say the product owner says "OK guys, I respect your time estimate of 6 story points, but I'm sure you can do some kind of quick-fix for this in half the time if you just put your mind to it."

Aha! He is trying to use internal quality as a variable. How do I know? Because he wants us to reduce the estimate of the story without "paying the price" of reducing the scope. The word "quick-fix" should trigger an alarm in your head...

And why don't we allow this?

My experience is that sacrificing internal quality is almost always a terrible, terrible idea. The time saved is far outweighed by the cost in both short and long term. Once a code base is permitted to start deteriorating it is very hard to put the quality back in later.

Instead I try to steer the discussion towards scope instead. *"Since it is important for you to get this feature out early, can we reduce the scope so that it will be quicker to implement? Perhaps we can simplify the error handling and make 'Advanced error handling' a separate story that we save for the future? Or can we reduce the priority of other stories so that we can focus on this one?"* (Kniberg 2007)

The teams must protect the developments and always make clear to the Product Owner the cost of quick-fixes and short term solutions.

Technical Backlog

The company should have an official technical backlog, which would contain issues/tasks concerning product improvement, like revision of quick/easy choices done, that can be done in a more productive way (with more performance per example).

One of the ideas is that in every Sprint, the Team choses one issue/task, from the technical backlog to implement/restructure. This type of tasks would include requests for managing system maintainability and other technical debts, like facilitate refactoring or making it more dynamic.

Even old code, when being modified should be continuously improved and refactored so the technical debt stops increasing.

The issues of the technical backlog cannot be always low prioritized, we need to guaranty that we will always work a bit for the technical backlog, it can be by choosing an issue from the technical backlog to do in a Sprint, or saving a percentage of the Sprint's time to attend to them. (Nikitina and Kajko-Mattsson 2011)

In the company's reality I advise that in every Sprint a task from the technical backlog is chosen to be included in that Sprint backlog, that should correspond to 10% of the Sprint's planned capacity.

4.6.4 Testing

There are two versions of testing that will be focused: Tests done by developers; and tests done by the QC team.

Developers testing

Although it is be very important to quality that personal and peer review always happens, testing cannot be set aside. Developers should deeply test their developments.

We need to have more unit testing, this needs to be an incremental process, like for every new functionality that we start we create unit tests, and when we need to change an existing functionality, we create unit tests.

The developer should test the code in his machine, in the development environment and extensively in the integration test phase.

Creating unit tests for integration, if it is possible would be a time saving measure.

QC Testing

It would be very good that the testing team accompanied every new feature, so they can develop test cases and test plans while the development team is developing, so it does not extend or delay the process. This way when the development is complete from the development team, and before it's pulled for testing, the tester would have a test case ready or almost ready to pull the new development.

The Kanban board would be helpful to view the stage of the feature, however it would be very important that the tester, or the team leader of the testers team participates in the Scrum meetings of the development team.

We probably will not need to affect a tester to each team, but a tester to each new functionality would be interesting.

"Presently, most of the test cases are being written during development, which has substantially reduced the feedback loop between the testing and development teams, and thereby, the overall testing time and effort." (Nikitina, Kajko-Mattsson, and Strale 2012)

4.6.5 Training

We need more training, more Lessons Learned sessions, and more good practices training sessions. This is extremely important to improve process and product quality, to support the introduction of process changes and to brief the new hires about the development method used. (Nikitina, Kajko-Mattsson, and Strale 2012)

These sessions can be made on an as needed basis for the purpose of supporting undergoing changes, or to share new information of good practices. Even if it is not created the habit of doing these sessions periodically they must be done often.

In a first stage I would suggest that a lessons learned session is performed once a month, so teams can share their lessons learned in the past two Sprints with other teams.

They might not be needed so often in the future, but they should never be put aside, not only they are important in terms of quality and code standards as they are motivational.

This should be short sessions of one hour, so focus is always present and so that sessions do not become a burden or even bring the feeling of being a waste of time.

4.7 Summary

In the table below – see Table 2 – is detailed a match between known difficulties reported in the diagnosis and actions that are proposed to reduce the impact of those difficulties:

Table 2 - Mapping between diagnosed difficulties and proposed solutions

Diagnosed Difficulties	**Proposed Solutions**
Communication	Scrum Ceremonies
Impact Analysis	Sprint Planning; WBS
Cross-functional resources	Sprint Planning; Sprint Backlog and Goal, working as a team to achieve it;
Evaluation	Daily Standups; Sprint Review
Trust	Transparency – Scrum
Team's cohesion	Scrum; goals and objectives
Motivation	Having a plan, a goal, a path to follow; Project ownership; teamwork
Changes in priorities	One Product Backlog; Kanban board; Goals definition; Two-weeks Sprints
Estimations	Sprint Planning; WBS;
Task-switching	Goal definition; Kanban board; WIP limit;
Quality	Scrum Ceremonies; WBS; Double check; Technical Backlog; Lessons Learned; Training

Chapter 5

Pilot Implementation

The underlying goal of this dissertation is to create a successful methodology, which satisfies the company's specific goals of software development, this topic describes the pilot implementation of such methodology. Upon the success achieved with this pilot, the proposal created in this work will be expanded to all the teams in the company, providing a better work environment for all the knowledge workers.

The pilot project was applied to three very different development teams. The main idea behind the pilot, is to evaluate the proposal to understand if it really fits the company's necessities and if it resolves the inherent problems.

5.1 Teams

This hybrid methodology was applied to 3 different teams and at the time of writing this dissertation, the teams ended their 4th Sprint, thus concluding the planned pilot time-frame.

The teams differ in some aspects:

- Team A: It is composed of 5 members: two back-end developers, 2 front-end developers and the team leader that has significant more experience in front-end.
- Team B: This is a content team, and it is composed of 4 members: 3 content creators, without technical background, one of which is the team leader; and a content configurator which has technical background;
- Team C: This team is composed by two very different sub teams:

- Team C1: This is a team very similar to Team A, it is composed by 4 members: 2 front-end and 1 back-end developer. The team leader is a back-end developer also.
- Team C2: This team is only composed of two members that work on a very specific part of the product, and in two technologies very different from the other team members. We have one member for each technology, however one of the members has knowledge in both technologies.

As the proposal was something very new to the company and to the affected teams, the teams received a 4 hours training session, where I briefly explained the practical aspects of the proposal. The theoretical aspects were presented to everyone in the development in a session with Q&A. All of this was done with my supervisor's supervision.

The teams were accompany through the whole process by myself and supported in each role, meaning that I supported, coached, and guided the team. And in some tasks, like leading meetings and communication with the Product Owner, I've even performed the role of Scrum Master.

As one of the failure factors was communication between teams I've also made it part of my job to share the good practices that each team was applying throughout the Sprints with the other teams. Gathering lessons learned from retrospectives and sharing them across the teams.

In Team A, we felt the necessity of supporting the Product Owner with an analyst, since analysts have a more wide functional knowledge of the product and as so can clarify many of the doubts that arise during the Sprint Planning, especially regarding other teams' dependencies.

5.2 Changes in the proposal

It was proposed that the team inspection was done to code, however what was established was that only functional team inspection[6] was to be done. I accepted this change, because functional team inspection is better than no team inspection, and because it can still find some defects.

5.3 Encountered Difficulties

In this topic the more relevant difficulties encountered in the pilot implementation will be described and detailed to a point.

[6] Team inspection done using the product user interface, to verify that the planned alteration occurred.

5.3.1 Not Cross-Functional Teams

One of the major difficulties of this project was that teams were not cross-functional.

The short-term solution to this would be the Product Owners bringing the same quantity of work for both layers in each Sprint. Of course, that this never happened, and as so we had Sprints were one of the layers would be practically only debugging.

Also related to this was the fact that as teams were not cross-functional many tasks were assigned to one person even before the Sprint started. Of course this brought many difficulties in doing the Sprint Planning since we would have not only to pay attention to the Sprint's capacity as we would have to pay attention to the amount of job planned for a specific person, and thus the Sprint goal relying on one and only person.

This goes against many of the agile practices, however it was a compromise that had to be made, otherwise we would never started this project.

And in the end the truth is that some of the front-end developers, faced with this situation started to show some interest in back-end and even started to learn it. Thus being pioneers in the walk toward a brighter future, the future of cross-functional teams.

When the Sprint had fallen on one person, we tried to motivate the teams to support that person and try to share the tasks. This was not usually well accepted, we have some lone wolfs, but is something that the teams are understanding and improving.

5.3.2 Estimate in Non-Cross-Functional Teams

As teams were not cross-functional, in the beginning was difficult for all the teams to estimate. Knowledge workers had difficulties in estimating work for other layer.

In Teams A and C, this matter is almost completely overcome, they are starting to understand better the time that each issue takes for each of the layers.

Team B is evolving, but as the job of the configurator is so much different than the job of the content creators, this was a slower process that it is not finished yet.

5.3.3 Different Types of Teams

The teams were all very different not only in the things that they created as even in the way they did their job and how they organized themselves before the pilot, so the impact of the pilot will be very different from team to team.

In Team A the members would tell their team leader what they were doing daily.

Team B, besides being the only team in the pilot, and one of the very few in the Development teams that did "double-check", also has a very organized One Note, were they have a checklist for the main workflow, like creating, double-checking, configuration and testing.

Another particularity of team A is that, with the exception of one member, none of the other members has any background in teamwork, they came from very different fields like investigation, and more individual professions.

Team C is a new team, because before the beginning of the pilot it corresponded to 3 teams, one of two team members, with a very driven team leader, with training in team management, another team of 3 members, being that one of them was, through the whole course of pilot borrowed to another team, and finally a two member team, which corresponds to team C2.

Team C2 has yet another particularity, it started as a team of 3 members, but before the beginning of the pilot, the two most veteran members of the team left the company, leaving only one member. This is not an experienced member, since it is freshly out of college. The other member joined the team just at the begging of the pilot, meaning that in the first two Sprints this member was on training so his participation in the pilot was very small.

Team C1 was following a reactive approach and team C2 was more autonomous, since the product they work on was not in production yet. But both did not follow any methodology.

These observations regarding the teams are important because with them we will better understand the discrepancies in the results.

5.3.4 Focus Factor

Team A changed to a Focus Factor of 65% in the second Sprint, as requested by their SPM, but they were never bellow the guide line in the burndown down chart since that alteration. This might not be a consequence of the Focus Factor, but of the fact that, especially in this team, developments are too concentrated in individuals. In the last 3 Sprints the mainly developments of the Sprints were concentrated on specific knowledge workers.

Team B was the team more affected by the Focus Factor. Since the Product Owners did not have an organized Product Backlog in the first Sprint the team had too little work for what they were used to. So in the middle of the Sprint the team leader requested more items to the Product Owner. In the second Sprint we increased the focus factor to 60%, and in the 3rd Sprint we increased to 70%.

The problem was also that in this team, since they need client validation before testing and being able to consider a task as done, much of the time in one estimate would be for that stage after client's evaluation. Sometimes the client took a long time to respond and as so the team would have time reserved for tasks that might or not get done, so it would happen that they would not have anything to do, besides debugging. This was particularly demotivating to one of the members, because the team had a big task allocated to a member and the other member got bored and demotivated for only doing debug, as one could imagine

Of course that by the third Sprint the Product Backlog was really stacked of estimated tasks and this situation was improved. The team still feels they have less pressure than before so we will need to make some adjustments in the future.

In Team C since they have more work than capacity, we started and ended with 50% Focus Factor; it still seems appropriate and until they have more resources or when they stabilize Team's C2 product this seems to be appropriate.

5.3.5 Physical Board

We have some attrition with the board; some members because they were lazy to get up and change things on the board; other members were worried because they felt we were creating waste with post-its and paper, it was not ecofriendly.

But the truth is that the majority is accepting the board and understanding how much we can win with this physical interaction of moving tasks, and seeing them as done. Clearly see how many drop-ins we had in the Sprint, how much we achieved and so on.

In the first retrospective one member said that the fact of seeing so many things in the done column made him pleased with him and with the team.

Two of the Product Owners look at the board as a pool of information, for the team, and are starting to look at it as a pool of information for them. But at least one of the Product Owners is not yet convinced of the amount of information that he can get from the board.

Since this was a pilot project the company felt it was better to do a small investment on the boards, so instead a white board the teams had a large mockup board, and this has some advantages and some disadvantages.

The advantages were that as the board was of some type of paper, sticky notes stick to it very nicely, and as the boards were very light we could take them to the Sprint Planning and Retrospective meetings.

One of the disadvantages was that we could not draw and erase lines, so we had to use tape, and any alteration to the tape would damage the board. However, since we had some delay with the logistics of the boards, we used the cabinets as boards, and thus allowing us to change the columns, we had only "fictional lines".

Team A and B got their boards in the first Sprint, but team C only in the end of the 4th Sprint. We had no good ideas of how to stick the boards to the cabinets without damaging them so we put up the boards in the windows parapets and this became a disadvantage. This became a disadvantage because in team A the board is next to the problematic member, and he does not, ever, gets up in the Daily Standup.

As Team C did not have a board through almost the whole pilot I noticed that the fact that they were using the cabinets, was prone to more interaction, the board was closer to their line of sight when they were up, thus increasing the Team's interaction with it.

When Team C got their board, it was a bit different from the others, it was lighter, and it was made of some sort of foam, so the Team leader had the idea of using suckers to stick it to the cabinets. This was great, of course that sometimes one of the suckers unstuck and the board might fall, but usually is there, visible, usable and with the perfect height.

5.3.6 Rivalry between teams

As I said, it was part of my job to share the good practices that the teams adopted throughout the Sprints. However one of the teams that was evolving less, was never very pleased to receive that information.

They would usually assume a defensive and negative perspective towards the other team practices. Team A was very defensive towards the good practices that Team C adopted, and team B was always very reluctant when compared to teams A and C.

Team C on the other hand would always adopt the good practices used in team A and B, if they made sense to their reality. This is a team with a very good critical sense, and the team leader has knowledge in agile methodologies.

5.3.7 Product Owner

The Product Owners were, sometimes, a difficulty. As we decided that in the pilot Product Owners would be represented by SPM's, this was sometimes a difficulty, since they are usually very busy knowledge workers. It is crucial for a project success that a good Stakeholder Involvement exists, and as so, it is very important that not only the Product Owner has the knowledge that is expected from a Product Owner as he must make the time to share that knowledge with the Team, whenever is needed.

We had to delay the beginning of the pilot, because Product Owners were not ready to start.

In the first Sprints and in some cases still now, Product Owners did not share the Product Backlog with the teams in advance and thus when the teams get to the Sprint Planning meetings they had no idea of which items would be discussed.

Till the end of the 4th Sprint Product Owners do not have one only Product Backlog, they have their work organized through markets and not prioritized as one. This, as I said in the proposal, was a big challenge that was not yet overcome.

5.3.8 Fines System

The fines system was well accepted by the majority of the members however two members were not happy with the system.

One of the members was fine with the system till the day she got late, due to traffic and instead of just paying the fine as all the other members saw the situation as a personal persecution. Due to this situation Team B decide to abolish the fines system.

The other element was against fines from day one, however that only got clear to the team in the day he got late. Team A kept the fines system, but that member refuses to pay if he gets late.

As was said in the paper (Kniberg 2007) in teams where members do not usually get late, the fines system is not necessary. That is not the case in the pilot teams. Actually, getting late to meetings is in the DNA of the company as it is in Portuguese DNA. It is not uncommon that people make an effort to get on time, but the other attendees do not, so the next time that person will also get late, because everyone does it, it is a vicious cycle. Because of this, middle management gave the example and assumed the system as valuable, they use it in their meetings and it seems they are happy with it, we see everyone doing an extra effort to be on time.

I believe in the fines system and consider it an advantage, since it represents an incentive, in my opinion, to be on time.

5.3.9 Problematic knowledge workers

In this experience we had two very problematic knowledge workers, knowledge worker A and B, corresponding to knowledge workers that belong to team A and B, respectively.

Knowledge worker A

This was one member that refused to pay fines, even though the whole team was in favor. He saw no productivity gain in any of the Scrum meetings with the exception of the Sprint Planning. He refused to participate in the board, move post-its or put new post-is in the board.

He would always sit in the daily standups and when was his turn he would say something like: “I did nothing”, with a joker face. When answering to the question “how much time did you spend in this item” he would always answer “It is in Jira”. He would often check is computer during the meeting, sometimes with things not related with the meeting.

What we did to deal with this situation was to always try to keep him involved, we insist in the questions and we insist in asking him to use the board. Regarding the daily standups we need to change the place where the board is so he does not has a place to sit, or a computer to get distracted. However I’m afraid of his reaction to this change.

From what I was able to ascertain this is a very qualified member, but I think that he is not much motivated, or happy and it seems it is not related with the pilot. However it is important that me or management understands what is affecting this knowledge worker motivation.

Knowledge worker B

This was an extremely problematic member whenever she was called to attention, by her superior.

Just as a remainder, all of the issues that were raised during a Sprint were discussed in the Retrospective, things like why someone refused to pay fines. Our retrospective would always include the Product Owner because it was extremely important that they see with their own eyes the team's evolution and difficulties.

The first really difficult situation with this knowledge worker was when she got late, due to traffic, and refused to pay a fine. She was revolted with the fact of paying fines due to traffic, till that day she had nothing bad to say about fines.

She also stated that was unfair that her team had daily scrums at 8:15 while other teams have it at 8:30 or 8:45. This happens because I need to be present at all the meetings, and as so they need to be at different times. The member was not able to understand that the money from the fines was for the team only, and so it was not a question of fairness or unfairness. The fines where there as an incentive to be on time in the meeting and not something that had anything to do with the other teams of the pilot program. The Product Owner as her superior also stated that the Daily Scrum meetings time should not be confused with the beginning of office hours which starts at 8:00. With this situation the team decided to stop using the system.

During the 4th Sprint I went to conference Agile Portugal. From the many keynotes and talks that I saw, one caught my attention regarding communication and collaboration between teams, the talk "Building Empowered Teams" by Wayde Stallman. In his talk Wayde proposed a 3 minutes exercise to help teams communicate and collaborate more, the "yes and" exercise. This is based in one of the improv theater exercises and consists in a person saying a sentence, the next person says "yes and", adds something to that sentence, and so on. This type of exercise supposedly helps teams to communicate, listen and collaborate. So I proposed all the teams to do this exercise in their retrospective.

Team C loved the idea, team A gave it a try but in team B retrospective the team leader was a bit reticent regarding their need of doing this type of exercise, again team B being reluctant when compared with other teams. As so I explained that this team also had communication problems, and as an example I stated that it was very common that in Daily Scrums two members were chatting. Well this was the blooming of chaos.

Knowledge worker B assumed it was referred to her and started to lash out and discuss and point fingers and situations to every one of her team members and especially to her team leader, it was a very ugly scene. And totally unnecessary.

This is an important reminder of how difficult it can be to manage knowledge workers and how important assertiveness is especially in this type of management.

5.3.10 Team Leaders

I cannot attest to the certainty of the questionnaire results since, it does not belong to my area of expertise, but it showed that some team leaders had a leader profile and others did not.

This was clear in the pilot, as we could observe that some of the team leaders were fit for it others were not.

Team's A team leader is loved by his team and always tries to serve as an example. He defends the team and protects it. The only problem is that he has a problematic member and I do not believe he has any power to do something about it, or the capacity to motivate him.

Team's B team leader took a softer approach during the Sprints, letting me guide the Daily Scrums, but started to evolve, and now it is practically autonomous in managing Daily Scrums.

Team's C team leader has been a great Scrum Master, and he is very involved in the project, this was actually the only team leader that never forgot to take pictures of the board. His team also sees him as a leader and supports him at his role.

It is important to say that leaders are nothing without their followers and however in Scrum the Teams should be self-managing we are not at that point yet and we need team leaders, teams still need to mature and to learn.

5.4 Lessons Learned and Changes

In this topic I will present the lessons learned and other alterations that happened in the course of the pilot. This will include improvements to the proposal, as well as good practices adopted throughout the pilot.

During the Sprints the teams adopted some particular practices, some proposed by myself others by team members, which I will not enter in detail, but that I will briefly describe:

- Definition of "done" very clear, relating to the maximum number/criticality of defects that would be allowed in a feature to be considered as "done";
- Team C decided to do not only functional team inspection as team inspection to the code;
- Team C also decided to do team inspection to unplanned items, even in some bug corrections;
- Creating sticky notes for bugs resolution, to facilitate team inspection and to provide sense of fulfilment in the end of the Sprint, as all closed bugs would be visible in the board;
- From Sprint to Sprint teams A and C detailed their WBS more in the Sprint Planning;

- Identify very clearly and visibly the due date of the item if it was before the end of the Sprint;
- Identify very clearly and visibly the hotfix version, in the applied cases;
- Creating small sticky notes for sub-tasks;
- Identifying the person working on that item in the post it, using that person initials;
- Always having a stacked Product Backlog preferentially with estimated items;
    - Include tasks that did not fit in the Sprint in the board, with red sticky notes, so if the team runs out of work they can pick up those tasks, particular important since Sprints have different weights in terms of work for each layer, and members are no cross-functional.

5.5 Boards Evolution

In teams A and C, I reduced the capacity of the column team inspection as a mean to oblige the teams to do it. These teams also adopted the idea of using stickers to identify that items were on standby by no specific reason.

By the end of the 4th Sprint I proposed to teams A and C, that we add another horizontal line in the board, to separate bugs from drop-ins as it would not only be more fulfilling for the teams as it would paint a better picture of the quantity of drop-ins.

The pictures bellow (see Figure 6 and Figure 7) represent the main difference between the start and end of the pilot. Unfortunately due to confidentiality reasons we cannot observe the pictures of the boards, which would clearly represent the team's evolution.

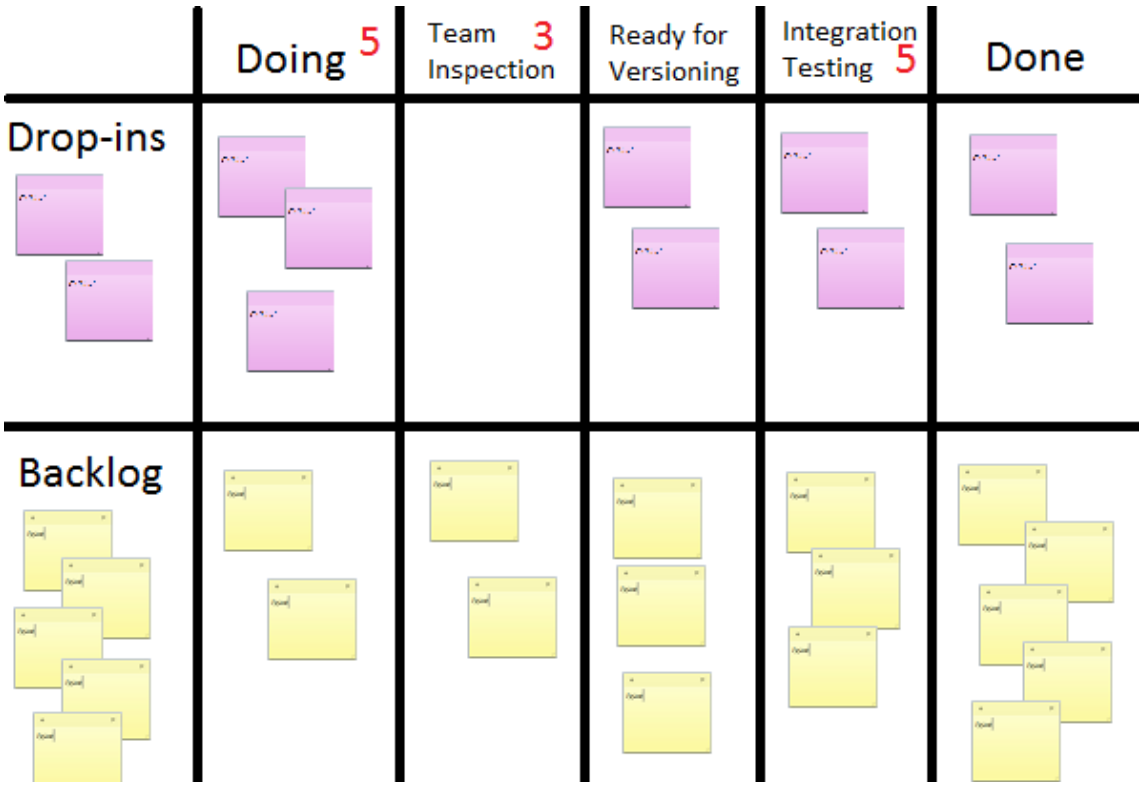

Figure 6 - Initial board of teams A and C, with the horizontal division separating backlog from drop-ins.

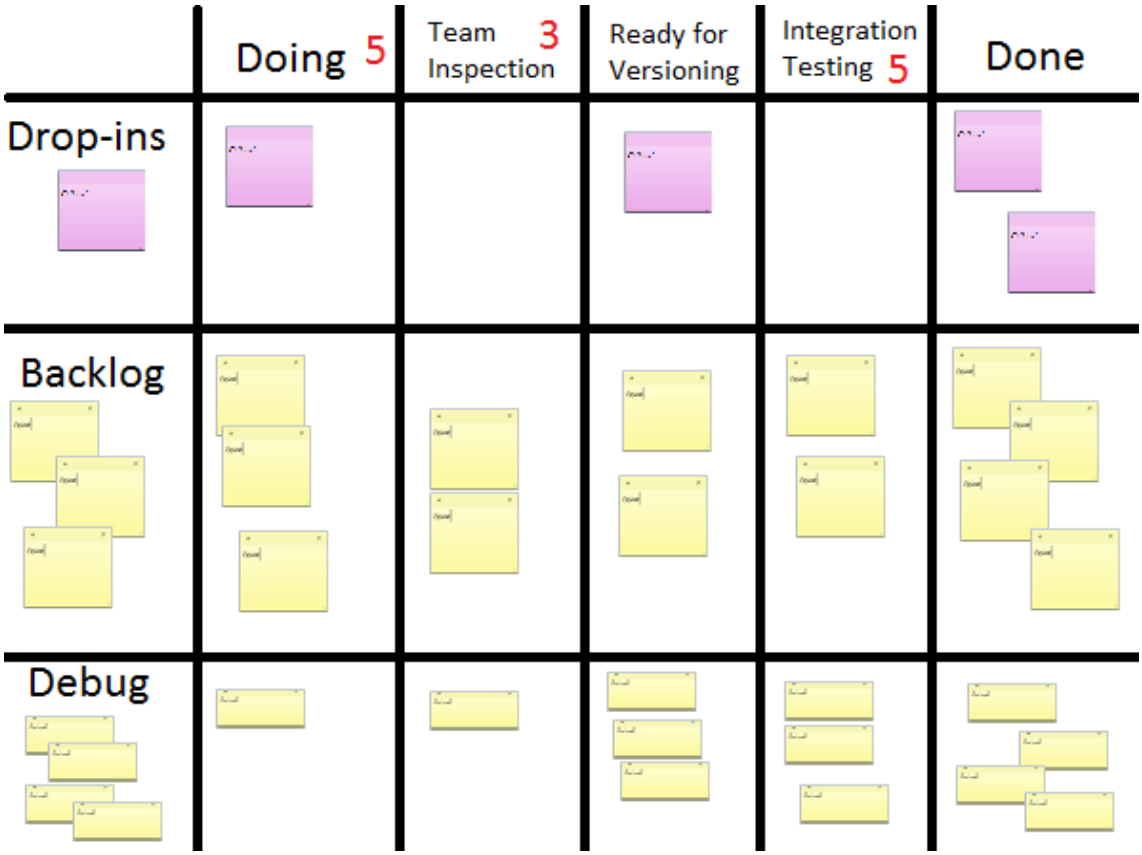

Figure 7 - Boards at the end of the pilot with an extra horizontal division for debugging

5.5.1 Team A

This team kept their board with the same columns through the Sprint.

This is, by far, the team that less interacts with the board, however the team leader is actively working to change that.

5.5.2 Team B

This team prefers to move each task to the doing column, which corresponds to creating content, but if they are doing double-check or integration testing they usually move the sticky notes to the doing column, thus making them have the capacity only in the doing column and in the configuration column.

This team has also another difference from other teams, usually everything that they create needs to get validated by the client, and although the client column acts as a buffer it is sometimes a bottleneck.

The fact that team B needs to have their items validated by the client is the main reason why their burndown chart never gets to zero, and why, by the end of each Sprint they do not have their Sprint Backlog completed. In the picture bellow – Figure 8 – we can observe Team's B Kanban board's columns.

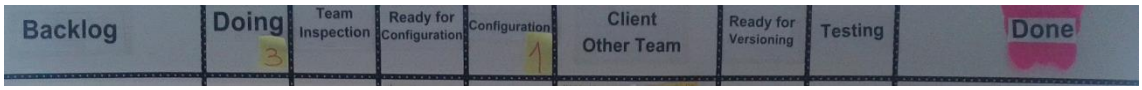

Figure 8 - Team B - Columns of the board: backlog; doing; team inspection; ready for configuration; configuration; client or other team; ready for versioning; testing; done

5.5.3 Team C

This team has good interaction with the board, and they are always very critic towards the board.

They keep discussing what works and not for then and in the end of the 4th Sprint they even considered separating the sticky notes in column "Ready for versioning" in to fictional columns, one that would correspond to the fact that a brunch was or no open. The other was to distinguish from their versioning and the versioning Team versioning.

Chapter 6

Analysis of Results

In this topic I'll be analyzing all the data that I was able to gather in the course of this project. I'll be referring the general opinions of knowledge workers that participated in the pilot, followed by subjective metrics, gathered from the first and second interviews and finally some of the objective metrics that I was able to gather from Jira.

6.1 Overcome challenges

One of the practices that I was afraid that would not be well accepted by management and knowledge workers was team inspection, however I was deeply wrong.

The teams are very pleased with team inspection. They saw how many defects they found in team inspection, that otherwise would have gone in the product till the Quality Control phase, which would represent a significant increase in Cost-of-Quality. They also said that team inspection was a great way to learn more and to learn more from another technology.

Another big achievement that I consider we achieved is that some of the front-end developers are doing some simple developments in back-end, doing some pair programing with the back-end developers and getting more and more autonomous. This resulting in an increased versatility of the team for back-end development.

We were also able to observe that they are a lot more focused and motivated than before, and this was not only clear for me as it was for the Product Owners that are also representative of the company's middle management.

One of the things that made this project possible was the support that it had from middle management. Middle management was not only the proponent of this project as supported it, always having a very present critical sense towards the proposal, but open to new ideas.

Although working with knowledge workers is a challenge it was a very thrilling experience. I had the luck of working with driven knowledge workers that trusted my judgment, but always thrived to understand why we would try this and that and why this was the way to go.

In general the knowledge workers that participated or that were somehow connect with the pilot project tried to put their skepticism a side and tried to believe in the project and in me. For this I'm forever grateful, this project would never became what it has, without them.

6.2 Subjective Metrics

In this topic the results regarding subjective metrics will be presented.

6.2.1 Knowledge Worker's Satisfaction

In the first round of interviews it was asked to the knowledge workers questions regarding their satisfaction according to the following factors: personal work; quality of his work; work of his team; quality of the work performed by his team and method of work. The interviewee had to choose a number between 1 and 10, where 1 was deeply unsatisfied and 10 deeply satisfied. The results of the 70 interviews are not relevant in the context of this dissertation, as so only the results of the participating members (15), before and after the pilot will be presented.

In the chart below (see Figure 9) it can be observed differences in average of their satisfaction before and after the pilot. This chart only represents the satisfaction of the participating knowledge workers in the pilot project.

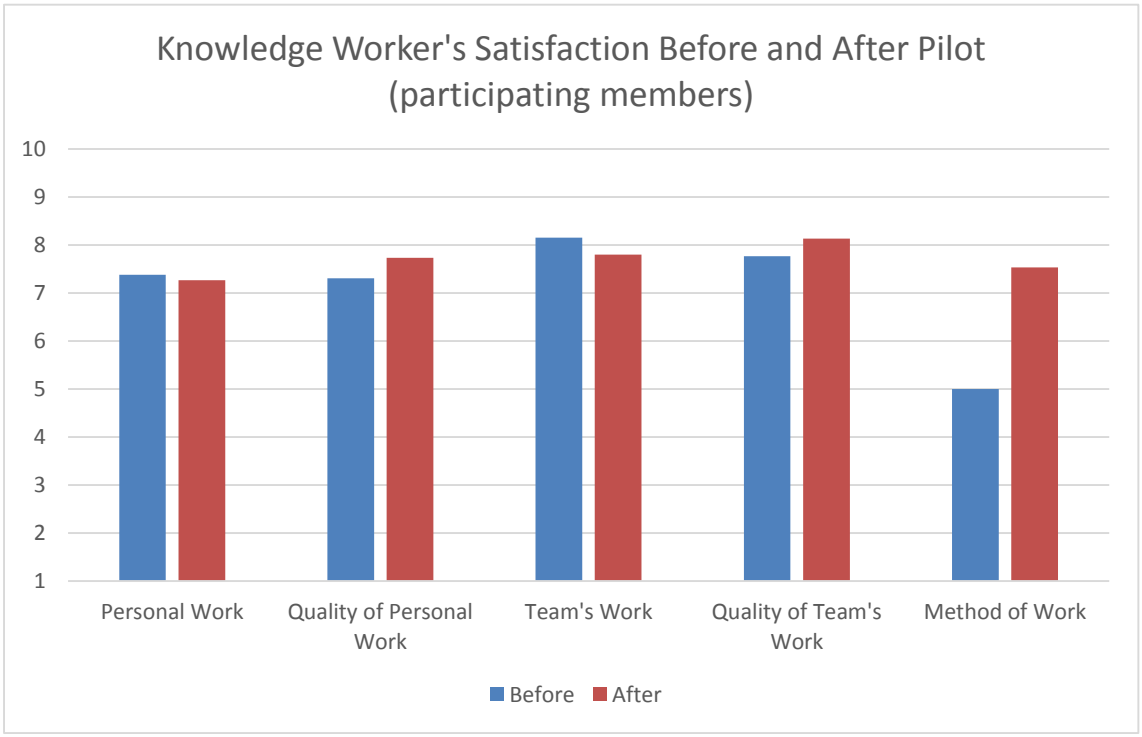

Figure 9 - Satisfaction of the knowledge workers that participated in the pilot project, before and after the project

It is important to refer that the majority of the knowledge workers did not remember which was his evaluation in the first phase, there was a gap of 4 months between the first and second interviews.

We can observe through the chart that there was a significant increase in the knowledge workers satisfaction with the method of work, and an increase in their satisfaction with the quality of their work and the quality of their team's work.

We can also say that in the first interviews knowledge workers might not trust me as they trust in the second ones and thus the results from the first interview might reflect a more optimistic approach than the reality. This meaning that however we cannot see, through the chart, more improvements, I'm certain they exist.

6.2.2 Pilot's Impact

As some time passed between the first and second interviews it was interesting to understand what improved of worsened with the implementation of the proposal. To assert this in the second interviews it was asked to the knowledge workers to evaluate the impact of the pilot on the following metrics: Task-switching; Multi-tasking; Focus; Planning; Work Method; Produced Quality; Satisfaction; and Motivation. To evaluate these metrics it was used a scale from 1 to 5, where: 1 – much worse; 2 – worse; 3 – same; 4 – improved; 5 – greatly improved.

The chart below (see Figure 10) represents the average of the knowledge workers answers regarding these metrics.

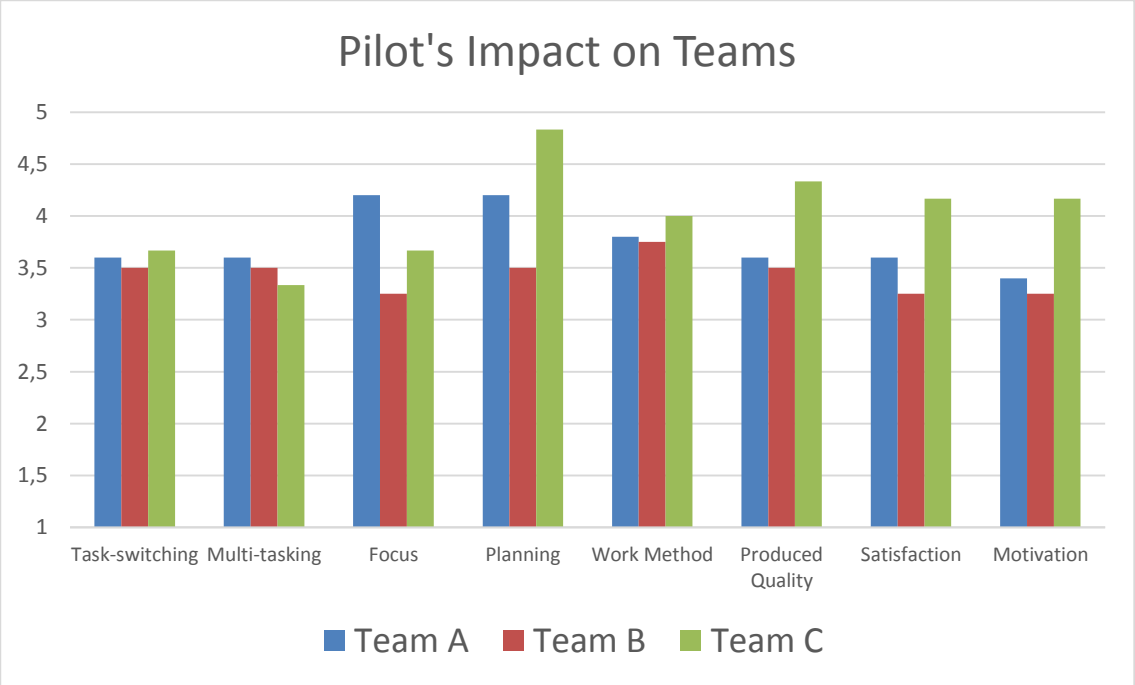

Figure 10 - Participating knowledge worker's perception of the pilot's impact

It is clearly visible that the pilot had a positive impact in all the aforementioned metrics. The average in all the metrics is above 3, which means, that in average, the project was an improvement in all areas.

We all know that average is something that is particularly influenced by numbers too high or too low, so average should not be the only form to represent data, but as this dissertation has a limited number of pages, the remaining information can be found in the Appendix B. In those charts (see Appendix B – Figure 15, Figure 16, Figure 17, Figure 18, Figure 19, Figure 20, Figure 21, Figure 22) it can be observed that for each of the points below an improvement occurred, meaning that the percentages represent the percentages of answers for each topic that were 4 (improved) or 5 (greatly improved):

- Task-switching – 58% of the aswers above 3;
- Multi-tasking – 42% of the aswers above 3;
- Focus – 67% of the aswers above 3;
- Planning – 92% of the aswers above 3;
- Work Method – 75% of the aswers above 3;
- Produced Quality – 75% of the aswers above 3;
- Satisfaction – 58% of the aswers above 3;
- Motivation – 61% of the aswers above 3.

I can objectively conclude that this project was success, and I know, since it was me that performed the interviews how each individual was affected by the project and what made him choose a two or a 5, but of course that information cannot be disclosed here due to the confidentiality with which the interviews where performed.

6.3 Objective Metrics

Objective metrics can have, of course, much more value in scientific investigations of this category, however in a period so short it is difficult to measure some of these objective metrics, so the results presented might not be inspiring.

6.3.1 Estimates

One of the objectives was that the teams estimated better. We put many actions into place to help with this situation, like doing WBS in the Sprint Planning and using Planning Poker to estimate.

There was no data before the pilot project, to support the study regarding this subject. During the pilot I collect some data to try to understand the degree of accuracy of the estimates performed by the team. Of course that I can never say that they estimate better than before in an objective way, since there is not any data to prove it.

Teams feel they estimate well than before, with the measures in place, team estimates, Planning Poker and WBS of tasks.

The times added and removed in the post-its at daily meetings corresponded to the times in Jira, so in this matter teams were accurate.

To evaluate the teams accuracy in estimating, for each team I made a comparison between their issues estimates and their logged times on Jira. The data in Jira is, of course, much more accurate than estimates, data in Jira can be measure to the minute and in estimates we used man-days. To this evaluation I used only complete tasks.

To measure the difference in man-days I rounded all the times in Jira to the closest man-day of that estimate, per example, if the issue had 10h of work this in teams A and C would correspond to 1,5 man-days, but in the case of team B to 1,25 man-days, as they use the quarter of a day. This is a necessary approach since we are asking teams to estimate in man-days and not in hours, so this conversion needs to be performed.

The charts below represent the weighted average of the ratio between Jira work logs and estimates for each Team, meaning that if a bar on the chart is above 100% means that the team underestimated the issue, and if a bar is below 100% means that the team spent less time developing the issue than they estimated.

We can observe, in Figure 11, some evolution with Team's A estimates, since we can observe that in the last 3 Sprints the estimates were less than 20% of the real time. In this team case, it is very visible the common oscillation in estimates

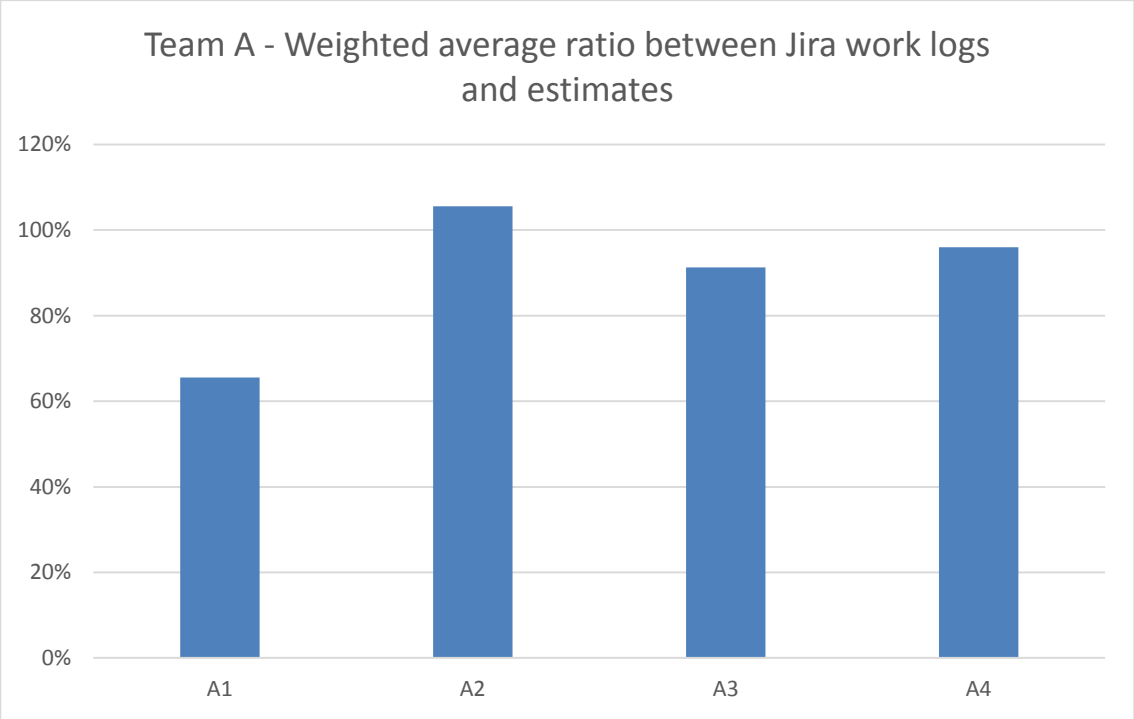

Figure 11 - Team A - Weighted average ratio between Jira work logs and estimates

We can observe, in Figure 12, that Team B underestimated very much in the first 3 sprints, much of this was particular to one big issue that gave this team too many complications, and in the last sprint they overestimated a big number of issues, but not by much, they estimated 18 issues for half-a-days work and each took around 2h. However we can still observe that in the 4th Sprint the Team had 100% of accuracy. From my work with this team, and all the data that I have the main issue here was that complex issue that was present through all the Sprints.

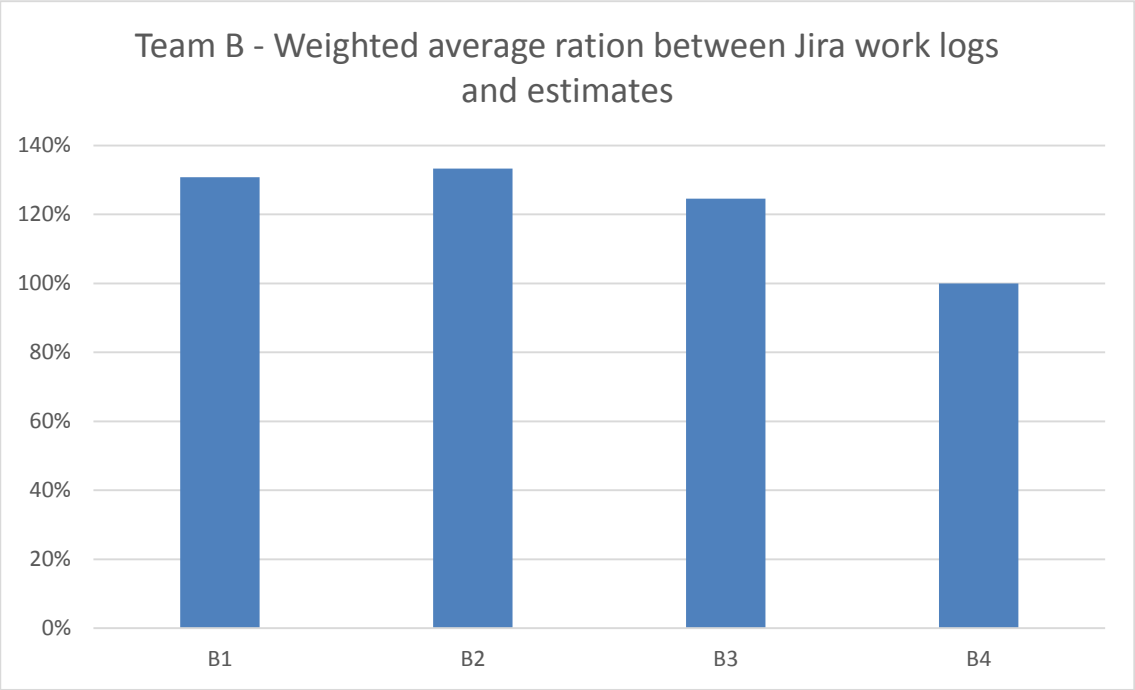

Figure 12 – Team B - Weighted average ratio between Jira work logs and estimates

In Figure 13, we can observe that Team C overestimated in the first Sprint and were more carefull in the last 3 Sprints, mainly because they got reluctant in the first Sprint, however their estimates are within the 20% limit, which means that they are estimating with some degree of accuracy, less than 10%, and we can tell that they are, in fact, evolving. In this team we can also see the so common oscillation in estimates.

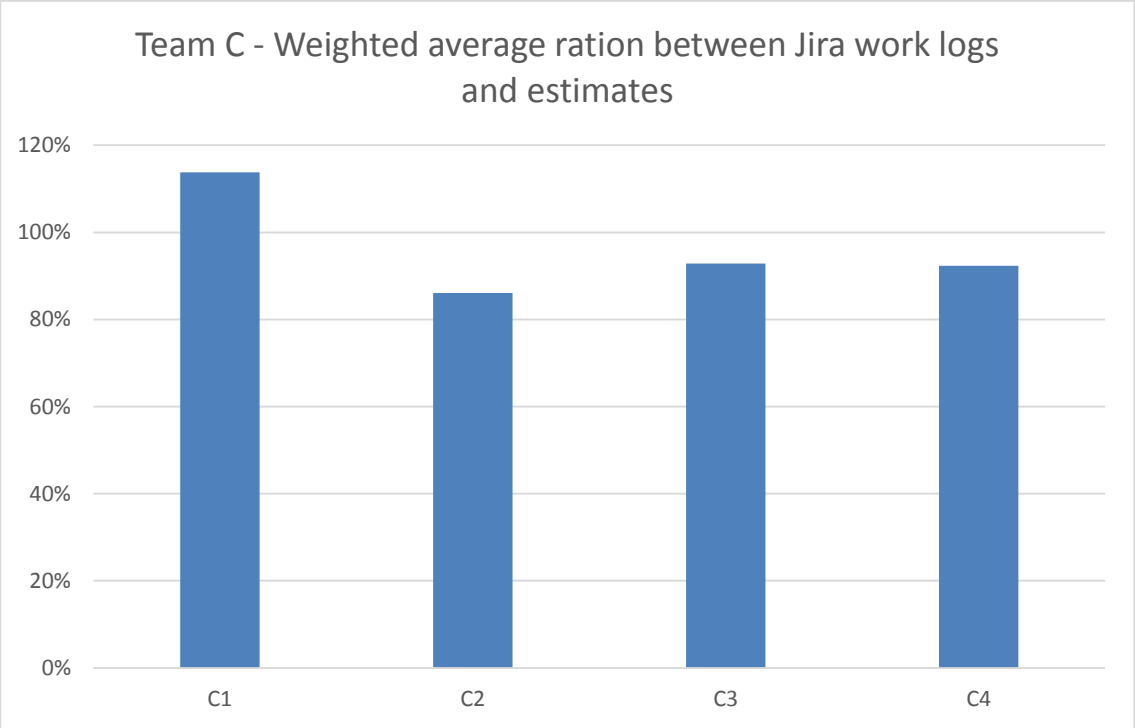

Figure 13 – Team C - Weighted average ratio between Jira work logs and estimates

6.3.2 Defects

With such a short period of time, around two months, we do not have any objective data that allows us to understand if we are improving quality or not. This type of evaluation requires more time to observe an impact.

Although we do not have objective data, we have subjective data, regarding the perception of the Team and Product Owners that allows us to conclude that we are improving the product's quality.

6.4 Validation and Analysis

Once again the validation is dependent on people.

At this moment in time, this project had the ultimate validation, management decided to put 3 more teams in the pilot project, meaning that they believe in its benefits.

I also had great feedback from the participating teams and from the Product Owners, and even much enthusiasm from other teams. So this is the best validation that a project like this could have.

After four Sprint Retrospectives, and with the individual interviews that I conducted with all the knowledge workers directly involved in the pilot, I was able to gather some subjective results regarding the impact of the methodology.

The points below describe what the majority of the team members feel changed with this project:

- Teams feel more focused on the tasks at hand and as so they have significantly reduced the amount of task-switching;
- Teams are more willing to test their implementations because they feel the pressure to show their work to the Product Owner;
- They assume that they work more as a team, they discuss their ideas instead of closing themselves;
- They feel they are more organized and that allows them to produce better software;
- They feel that they have a plan to follow;
- They assume that now they think more and verify their work;
- They say that Daily Scrums are very good because now they know what everyone did;
- They understand the importance of doing WBS in the Sprint Planning – they evolved during the four Sprint Plannings, detailing the WBS more. "We now have a map, we can see if we are late or not";

- They see team inspections as valuable;
- All the Scrum meetings are also seen as valuable;
- They feel that now things are put in perspective;
- They feel their team communicates more;
- They like to see the evolution of the items on the board;
- They like to have a Team objective – Sprint Backlog and Goal;
- They see as something very positive the fact that is the team that does the estimates and not only one person, and they feel the importance of committing to it;
- They believe that drop-ins are a lot clearer for the Team and Product Owner. "Tasks are more defined and objective, now we know what to do";
- The teams feel they have a better understanding of what is planned;
- They assume that now they estimate taking into account testing and team inspections, thing that they did not do before, they pay more attention to quality;
- They say this project improved their motivation;
- They feel they are more autonomous;
- Overall they feel that this proposal is a positive thing!

Chapter 7

Conclusions and Future Work

On implementing a hybrid methodology of Scrum, Kanban and other good practices like team inspection we are able to subjectively conclude that we successively reduced the amount of task-switching of the knowledge workers through the definition of capacity and plans/goals to keep the teams more focused and less disperse prone.

Scrum with its meetings presented itself as a great "tool" to improve communication between team members and between the Team and its superior.

The choice of using the middle management as Product Owners presented itself as a great form of assuring some organizational support to the process and to the experience.

The team inspection proved itself as good practice to reduce the Cost-of-Quality as to increase the knowledge workers' knowledge.

My last and most important recommendation is that Agile does not suggest you to define common sense, nor it tells you to ignore disciple, or even to do not produce documentation.

Agile is not chaos, nor anarchy! Agile foundations are constantly improve, inspect and adapt, do lessons learned, remove impediments or worthless processes. I actually believe there is such a thing as too much agility, but that is not being Agile, that is being chaotic!

Regardless of you implementing Scrum, XP, Kanban, in no place is said you should never question the methodology, or follow it just like a prescription. Agile is much more than a recipe, is constant learn and improvement, is constant growth, it is much more than a simple set of rules.

7.1 Objectives Satisfaction

The teams are more motivated and happy with their work, they fill the sense of commitment and they feel they have establish goals and what route to take.

This commitment is so clear that now, instead of the Product Owner harassing the Team to deliver is the team that harasses the Product Owner to remove the impediments. Showing how much the Team is committed to deliver the items on time.

"Quality work is not done by mistake, and it is not done by unmotivated or disgruntled employees. The developers must care about their work, strive to consistently improve, and be proud of what they produce." (Humphrey and Over 2010)

We can clearly observe, through the subjective metrics that this project was a success, it brought and improvement in all the areas that the proposal wanted to improve. No question for communication was need because it was very clear that, at least inside the team, it was severely improved.

We can conclude, that this project was a success, and the objectives satisfied since management decided to expand this pilot project to 3 other teams, one of them being the biggest and more difficult team.

It is also important to refer that from the 15 knowledge workers, corresponding to the team's members only one did not see the gain in the Scrum meetings, with the exception of Sprint Planning, we can perfectly state that this methodology works, brings benefits and value to the organization starting with its core, its knowledge workers.

7.2 Future Work

As any agile implementation we can never say: "My work here is done.". There are always new improvements to do and adjustments to be made. In this case there are some challenges that are already very clear to me.

7.2.1 Challenge #1 – Evaluation

Nothing became clearer during this pilot as the fact that these knowledge workers need evaluation.

What we were able to observe in some situations, especially with the problematic members is that knowledge workers at this company are used to not having to proof anything, or give any satisfaction. Are not used to be drawn to attention, and overreact when they are. This clearly shows the need of a fair evaluation system, which cannot be totally objective nor totally subjective. We observe the impact that we got with Scrum and especially the daily meetings as it became a source of information were the team can evaluate itself and its members.

7.2.2 Challenge #2 – Analysis

Having the analysis walk in pair with the development is something that we were not yet able to overcome.

The most critical example of this real problem was observed during Sprint 4 in Team A.

Team A had 3 tasks of the issue X planned for Sprint 4, this was a Sprint with 11 and a half business days. The main particularity of this issue was that it could not start unless analysis was done since the development team needed a worksheet created by the analysis team. This issue was on stand-by during 7 business days waiting for analysis making it impossible for the team to deliver what they committed to in that Sprint. This was not the only time this happen but was the most critical one.

This is a planning problem that we will attend to outside the scope of this dissertation, because if we only considered analyzed items in the Sprint, analysis would not be a problem.

However, implementing the proposal in the analysis team will probably help with the planning and thus reducing this difficulty.

7.2.3 Challenge #3 – Product Owner

The third challenge is that at the moment, with only 3 teams in the program we see that sometimes it can be difficult to schedule the Scrum meetings, due to the Product Owner's busy schedules, as so we'll need to find a solution, as in a first stage we will be expanding the pilot program to 3 other teams, making a total of 6 teams using this methodology.

There are some measures that I already suggested for solving this problem, however they were not tested yet.

One solution would be for the SPMs to share their role as Product Owners with the analysis team, making them proxies. However, this type of action requires that the Product Owner's pass on the responsibility of making priority decisions to the proxy, otherwise it will not work.

The other solution would be to create a new role in the company for a more specialized Product Owner. This would be a person that would have the knowledge and the authority to adjust priorities and its main objective would be to act solely as a Product Owner.

7.2.4 Challenge #4 – Sprint Retrospectives

In the pilot program retrospectives were longer than optimal, this was actually something pointed out by management. There are some reasons for this:

- Much of my evaluation of the pilot's impact was done in the retrospectives, this was where I would ask questions to the team about the process, see Events;

- The teams felt empowered to talk, they felt they now had a change that they never had before, so sometimes they would focus the same points that have been impediments for the teams for some time, like hardware, incomplete analysis, and so on;
- As it was a pilot, and much of my evaluation depended on these meetings, I allocated a big time frame for the meeting, around 2 and a half hours.

With time, retrospectives will get shorter, because teams will get used to process, be more prepared for it and more focused on the topics. However what we decided was to reduce the Retrospectives to a one hour event.

With time not only teams, as management will better understand the benefits that come from retrospectives, but for now it is important that the length of retrospectives does not represent a failure factor for the project.

Actually in the 4th Sprint the retrospectives of teams A and C took less than one hour, only team's B retrospective took a bit more, but this was explained in the topic Problematic knowledge workers.

7.2.5 Challenge #5 – Cross-functional teams

Teams need to evolve to the point where they are cross-functional. This is a necessity, which I see that management already understood but that many knowledge workers did not.

I would say that the majority, if not all of the back-end developers, does not want to learn anything from front-end development, this might happen for many reasons but one is certainly the technology used in front-end, that is considered by many outdated.

That is the thing with technologies, they never stop evolving. The company is changing the product to a new technology and maybe then the back-end knowledge workers become less reluctant to learn front-end. Only time will tell.

This behavior changes a bit when we look to the front-end developers. At the moment 2 front-end developers are able to do some back-end development, and another is on the same path. As the front-end technology is considered outdated, many of the front-end developers that were in my interviews, share how unmotivated they were with it. Because as someone said "If I look to my curriculum it says x years of experience in a "dead technology"". So they see this learning process as motivational, till the product migrates to the new technology.

7.2.6 Challenge #6 – Pair Programming

In observing teams at work, I saw that sometimes some knowledge workers do a bit of pair programing understanding how important it can be in some tasks.

What I think we should do next would be implementing pair programing in some specific situations, like new members joining the team, knowledge workers learning a new technology, and most importantly to share knowledge between knowledge workers, since at the moment the company depends highly in individual human beings.

However the company has already started to incentive this knowledge share, promoting passaging of knowledge between two content teams, which was a small step for a great future.

7.2.7 Challenge #7 – Operations Teams

As this project evolves, and regardless of the company's choice of expanding it to all of the teams, I see a challenge regarding the Operations Teams.

Their work is a lot more reactive than the work done by development teams and as so this proposal might not work for them, however it needs to be evaluated, since my understanding of how Operations Teams work is very small.

But the challenge here would be to find a proposal for the Operations team, which in a first look would be more Kanban than Scrum. It would probably be a very detailed Kanban Board, and Daily Standups and Retrospectives. This is of course a hypothesis, which needs much further evaluation.

7.2.8 Challenge #8 – Estimations and Workflow

One of the difficulties that we felt was related to Team B, due the workflow of their work. As they create content that needs client's validation to advance, this is clearly a bottleneck that influences estimates.

We feel that the team does not have enough pressure to carry on as they were used too, because of this, they might feel that everything is in control and the Product Owner feels that their productivity has reduce because of this.

We need to evaluate this situation in further detail and try to understand what can be done:

- Increasing Focus Factor;
- Creating queues for planned work, so they have a clearer view of what needs to be done;
- Change from man-days to hours, since they were the team that felt the need to estimate to ¼ of a day.

In the next Sprint we will start with the first option and increase the Focus Factor to 80%, and inspect and adapt.

7.2.9 Challenge #9 – Task-switching and Multi-tasking

Teams still do more task-switching than would be advised, but with the extension of the pilot to other teams, and thus improving the alignment between teams, it is expected that teams reduce the need of task-switching.

I also think that we need to analyze in further detail the columns capacity, because it seems to me that teams abuse the usage of the standby stickers.

7.2.10 Challenge #10 – Tests

At this moment we have not included any tester in our meetings, but we are going to change that.

Team A is doing a big development and it was planned that a tester was assigned to test the feature in development. In the end we will, probably, include the tester in the Sprint Review, and that will represent an improvement.

Of course, that the future is including testers, at least, in Sprint Plannings, so they can help see interactions, and prepare their test cases with full knowledge.

7.2.11 Challenge #11 – Documentation

Documentation was not part of the scope of the proposal, however we included in the Sprint Backlog, when relevant, tasks and estimates for creating the documentation that in some cases was the developer's responsibility.

There is, of course, much more to be done regarding documentation and that is the challenge.

References

Anderson, David J. 2010. Kanban Successful Evolutionary Change for Technology Organizations.

Beck, Kent, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. 2001. "Manifesto for Agile Software Development." http://agilemanifesto.org/.

Berkun, Scott. 2008. *Making Things Happen*: O'Reilly Media.

Boehm, B., and R. Turner. 2005. "Management challenges to implementing agile processes in traditional development organizations." *IEEE Software* 22 (5):30-39.

Budau, Tiberiu M., and Pascal Bihler. 2012. Kanban in a nutshell. University of Bonn.

Carvalho, W. C. D., P. F. Rosa, M. D. Soares, M. A. T. da Cunha, L. C. Buiatte, and Ieee. 2012. "A comparative Analysis of the Agile and Traditional Software Development Processes Productivity." In *2011 30th International Conference of the Chilean Computer Science Society*, 74-82.

Czerwinski, Mary, Eric Horvitz, and Susan Wilhite. 2004. "A diary study of task switching and interruptions." Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Vienna, Austria.

Davies, Rachel, and Liz Sedley. 2009. *Agile Coaching*: Pragmatic Bookshelf.

Ferguson, P., W. S. Humphrey, S. Khajenoori, S. Macke, and A. Matvya. 1997. "Results of applying the personal software process." *Computer* 30 (5):24-31.

Garzas, J., and M. C. Paulk. 2013. "A case study of software process improvement with CMMI-DEV and Scrum in Spanish companies." *Journal of Software-Evolution and Process* 25 (12):1325-1333. doi: 10.1002/smr.1605.

Garzás, Javier, and Mark C. Paulk. 2013. Can Scrum help to improve the project management process? A study of the relationships between Scrum and Project Management process areas of CMMI-DEV 1.3. .

Grojean, C. A. 2005. "Microsoft's IT organization uses PSP/TSP to achieve engineering excellence." *CrossTalk* (3):8-12.

Group, The Standish. 2014. CHAOS. edited by Project Smart.

Hansen, M. T., and H. Baggesen. 2009. *From CMMI and isolation to Scrum, Agile, Lean and collaboration*. Edited by Y. Dubinsky, T. Dyba, S. Adolph and A. Sidky, *Agile 2009 Conference*.

References

Hilburn, Thomas B., and Massood Towhidnejad. 2000. "Software quality: A curriculum postscript?" *SIGCSE Bulletin (Association for Computing Machinery, Special Interest Group on Computer Science Education)*:167-171.

Humphrey, W. S. 2007. "Software process improvement - A personal view: How it started and where it is going." *Software Process Improvement and Practice* 12 (3):223-227. doi: 10.1002/spip.324.

Humphrey, W. S. 2008. "The process revolution." *CrossTalk* 21 (8):4-6.

Humphrey, W. S., and James W. Over. 2010. *Leadership, Teamwork and Trust: Building a Competitive Software Capability*. Edited by Pearson Education.

Ikonen, M., P. Kettunen, N. Oza, and P. Abrahamsson. 2010. "Exploring the Sources of Waste in Kanban Software Development Projects." Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference on, 1-3 Sept. 2010.

Kandt, RK. 2002. "Organizational Change Management Principles and Practices."

Kniberg, Henrik. 2007. *Scrum and XP from the Trenches: Enterprise Software Development*: Lulu.com.

Kniberg, Henrik. 2010. *Kanban and Scrum - making the most of both*: Lulu.com.

Kniberg, Henrik. 2011. *Lean from the Trenches: Managing Large-Scale Projects with Kanban*: Pragmatic Bookshelf.

Koch, A. S. 2005. "TSP can be the building blocks for CMMI." *CrossTalk* (3):4-7.

Nikitina, N., M. Kajko-Mattsson, and M. Strale. 2012. "From scrum to scrumban: A case study of a process transition." Software and System Process (ICSSP), 2012 International Conference on, 2-3 June 2012.

Nikitina, Natalja, and Mira Kajko-Mattsson. 2011. "Developer-driven big-bang process transition from Scrum to Kanban." Proceedings of the 2011 International Conference on Software and Systems Process, Waikiki, Honolulu, HI, USA.

Pink, Daniel H. 2011. *Drive: The Surprising Truth About What Motivates Us*: Riverhead Books.

Potter, Neil, and Mary Sakry. 2011. Implementing Scrum (Agile) and CMMI Together.

Rickets, C. A. 2005. "A TSP software maintenance life cycle." *CrossTalk* (3):22-24.

Rubin, Kenneth S. 2012. *Essential Scrum: A Practical Guide to the Most Popular Agile Process*: Addison-Wesley Professional.

Rutherford, Kevin, Paul Shannon, Craig Judson, and Neil Kidd. 2010. "From Chaos to Kanban, via Scrum." In *Agile Processes in Software Engineering and Extreme Programming*, edited by Alberto Sillitti, Angela Martin, Xiaofeng Wang and Elizabeth Whitworth, 344-352. Springer Berlin Heidelberg.

Salvucci, Dario D., Niels A. Taatgen, and Jelmer P. Borst. 2009. "Toward a unified theory of the multitasking continuum: from concurrent performance to task switching, interruption, and resumption." Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Boston, MA, USA.

Sanchez-Segura, M. I., J. García, A. Amescua, F. Medina-Dominguez, and A. Mora-Soto. 2008. "A study on how software engineering supports projects management." Bridgeport, CT.

Schwaber, K., and J. Sutherland. 2013. The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game. Accessed January 24th, 2014.

SEI. 2010. CMMI® for Development, Version 1.3 - Improving processes for developing better products and services. Carnegie Mellon University.

Sinclair, B., C. Rickets, and B. Hodgins. 2011. "Developing an accurate, reliable method." *CrossTalk* 24 (4):25-31.

Speier, Cheri, Joseph S. Valacich, and Iris Vessey. 1997. "The effects of task interruption and information presentation on individual decision making." Proceedings of the eighteenth international conference on Information systems, Atlanta, Georgia, USA.

Sturdy, Andrew, and Christopher Grey. 2003. "Beneath and Beyond Organizational Change Management: Exploring Alternatives." *Organization* 10 (4):651-662.

Teixeira, Bruno Afonso. 2013. "Agile and Traditional Project Management: bridge between two worlds to manage IT Projects " Master masterThesis, Engineering School, University of Minho (29113).

References

Tonini, A. C., M. M. de Carvalho, and M. de Mesquita Spinola. 2008. "Contribution of quality and maturity models to software process improvement." *Contribuição dos modelos de qualidade e maturidade na melhoria dos processos de software* 18 (2):275-286.

Appendix A

Interviews

The original questionnaire can be found in the following link: http://paginas.fe.up.pt/~ei07157/assets/eu_tu_equipa.pdf

A.1. Dr. Luís Graça questionnaire – in Portuguese

EU, NÃO SOU PERFEITO, TU NÃO ÉS PERFEITO, MAS A NOSSA EQUIPA PODE SÊ-LO!

Objetivos

Identificar o principal papel funcional que você habitualmente desempenha na sua equipa.

Conhecer a estrutura e a dinâmica das equipas.

Desenvolver a capacidade de avaliação da eficácia das equipas.

Procedimento

O questionário (1), que a seguir vai responder, é constituído por 56 itens ou proposições, distribuídas por 7 secções, identificadas pelas letras do alfabeto que vão de A a G. Dentro de cada secção, os itens estão numerados de 1 a 8 (por exemplo, A1, A2, B1, B2).

Na coluna do seu lado esquerdo, comece por assinalar (com um círculo ou uma "bolinha") 1,2 ou 3 (no máximo!) dos 8 itens ou proposições que compõem cada uma das 7 secções: os que escolher devem traduzir ou descrever o melhor possível o seu comportamento mais característico ou frequente quando você faz parte de equipas ou grupos de trabalho.

Por analogia com uma orquestra, pode-se dizer que há um instrumento musical que você toca melhor (neste caso, há um papel que você desempenha com mais frequência, por gosto ou obrigação).

A seguir, na coluna do seu lado direito, correspondente à proposição ou item que você assinalou (até um máximo de 3, por secção!), irá dar uma pontuação, usando para o efeito uma escalda de 1 a 20 valores. O subtotal, por secção, deve perfazer exatamente 20 valores pelo que o total será igual a 140 (ou seja, 20x7). Você é que deve, obviamente, distribuir os 20 pontos por cada um dos itens que assinalou. Por exemplo, se o "instrumento musical" que melhor domina é o violino, mas também arranha o violoncelo, ao primeiro poderá dar os dezasseis pontos e ao segundo os restantes quatro (16+4 = 20). Repare: não se trata de assinalar o papel (ou o "instrumento") que você gostaria de desempenhar (ou tocar), mas sim aquele que efetivamente desempenha (ou toca), pior ou melhor.

A – Quando estou envolvido num projecto com outras pessoas:		
A1	Fico descansado quando vejo que o trabalho que tem de ser feito está **devidamente planeado, estruturado e organizado**	
A2	Sou especialista em detectar **erros e omissões** de que os outros, em geral, não se apercebem ou não dão conta	
A3	Reajo vigorosamente quando me apercebo de que estamos a afastar-nos do **objectivo principal** de uma reunião de trabalho	
A4	Sou uma pessoa com perspicácia para descobrir **as ideias e os progressos mais recentes** num determinado campo de conhecimentos ou de aplicações	
A5	Faço uma **análise objectiva das ideias dos outros**, pesando os respectivos prós e contras ou identificando os respectivos pontos fortes e fracos	
A6	Tenho uma tendência natural para **coordenar as pessoas e as tarefas do grupo**	
A7	Costumo sobretudo **dar ideias ou fazer sugestões originais**	
A8	Estou sempre pronto a **apoiar as ideias ou sugestões** que se me afiguram úteis para solução de um problema	
Subtotal		20

B –Procurando obter satisfação através do meu trabalho:		
B1	Gosto sobretudo de **influenciar as decisões** que são tomadas no seio do grupo	
B2	Sinto-me particularmente bem quando o trabalho requer um **elevado grau de concentração e atenção**	
B3	Preocupo-me sobretudo em **ajudar os colegas a resolver as dificuldades ou problemas** que enfrentam	
B4	Gosto de ponderar, de maneira crítica, as **várias soluções alternativas** que se apresentam para a resolução de um problema	
B5	Tenho um jeito especial para **conciliar diferentes pontos de vista e chegar a consensos** dentro do grupo	
B6	Aprecio muito em particular a possibilidade de **explorar diferentes perspetivas ou métodos** na abordagem e resolução de problemas	
B7	Estou mais interessado em **coisas práticas** do que em ideias ou teorias abstratas	
B8	Na resolução de problemas, tendo a adotar uma abordagem essencialmente criativa	
Subtotal		20

C – Quando a equipa está a tentar resolver um problema complexo:		
C1	Quaisquer que sejam as pressões(internas ou externas) sobre o grupo, mantenho com firmeza a minha **abordagem centrada no problema** e na sua resolução prática	
C2	Gosto de explorar ideias que possam ter uma **aplicação mais ampla** do que em relação à tarefa que temos em mãos	
C3	Costumo **mobilizar e utilizar**, de maneira produtiva, **as competências e os talentos dos outos**	
C4	Muitas vezes sou eu quem encontra **uma abordagem nova para um problema velho** que se arrasta no seio do grupo	
C5	Mantenho debaixo de olho aquelas **áreas de trabalho mais criticas** onde possam surgir inesperadamente dificuldades ou problemas	
C6	Antes de escolher a solução A,B, ou C, gosto de ter a minha disposição um **leque variado de hipóteses alternativas**, avaliá-las cuidadosamente e apresentá-las ao grupo	
C7	Sempre que necessário, sou capaz de **impor aos outros os meus pontos de vista pessoais**	
C8	Dentro das minhas possibilidades e limitações, estou **sempre pronto a ajudar a equipa**	
Subtotal		20

D – Na execução do meu trabalho diário:		
D1	Tenho tendência para **descobrir sistemas, padrões ou modelos coerentes** onde os outros habitualmente só vêem dispersos e desconexos	
D2	Estou particularmente atento para que **nada de vago ou de impreciso** possa existir em relação às minhas tarefas e objectivos	
D3	Não tenho problemas em **trabalhar com qualquer tipo de pessoas**, desde que elas possam trazer um valor acrescentado à equipa	
D4	Habitualmente sou capaz de **encontrar argumentos suficientes** para refutar as propostas que me parecem infundadas	
D5	Confrontado com ideias **que me parecem interessantes** (mesmo que venham de fora), faço questão de as explorar mais profundamente	
D6	Não tenho qualquer relutância, numa reunião de trabalho, em **enfatizar o meu próprio ponto de vista**, sempre que o julgue necessário	
D7	**Estar ocupado** é algo que me da uma verdadeira satisfação pessoal	
D8	Sinto não só necessidade como interesse em conhecer bem as pessoas com quem trabalho no meu grupo ou equipa	
Subtotal		20

E –Se de repente me for dada uma tarefa difícil, com um prazo apertado e um grupo de pessoas que não conheço (ou conheço mal):		
E1	Sinto muitas vezes a minha **imaginação bloqueada** ao trabalhar em grupo, debaixo de pressão	
E2	Acho que as minhas qualidades pessoais particularmente apropriadas para **negociar e obter um acordo** no seio do grupo	
E3	Raramente os meus **sentimentos pessoais** interferem com a minha capacidade critica	
E4	Aos olhos dos outros pareço revelar um **forte sentido de urgência**	
E5	Sou capaz de **trabalhar com pessoas muito diferentes umas das outras** (em termos de formação, experiencia, personalidade, origem social, sexo, idade, etc)	
E6	Sinto que, por vezes, vale a pena uma pessoa **correr o risco de ser temporariamente impopular** se, com isso, conseguir melhorar o desempenho do grupo	
E7	Habitualmente conheço **a pessoa certa no sitio certo**, cuja a experiencia ou cujos os conhecimentos são particularmente apropriados às necessidades do grupo	
E8	Esforço-me sobretudo por **criar uma boa estrutura organizativa**	
Subtotal		20

F – Quando de repente sou confrontado com a ideia (ou a necessidade) de me envolver num novo projeto:		
F1	Sinto-me feliz por **tomar a iniciativa** quando é necessário agir	
F2	Em primeiro lugar, **abordo e analiso o problema cuidadosamente**	
F3	Comprometo-me sobretudo em **mobilizar e envolver outras pessoas**, se for caso disso	
F4	Para mim **é difícil dar o meu melhor** no trabalho quando os objetivos do grupo não estão claramente definidos	
F5	Antes de pegar num novo projeto, preocupo-me sobretudo em **acabar o trabalho que tenho em mãos**	
F6	Em geral, sou capaz de **responder positivamente aos meus colegas**, às suas ideias, às suas propostas ou às suas iniciativas	
F7	Em muitas situações, sou capaz de **adoptar um ponto de vista independente e inovador**	
F8	Começo por **explorar, a minha volta**, a existência de possíveis oportunidades, ameaças, pontos fracos e pontos fortes (por exemplo, pessoas e outros recursos)	
Subtotal		20

G – Contribuindo, em geral, para projetos de grupo:		
G1	Tento sobretudo **influenciar positivamente os membros do grupo**, dentro e fora das reuniões de trabalho	
G2	As minhas **observações críticas** podem levar o seu tempo a serem elaboradas mas em geral acertam no alvo	
G3	Para o meu estilo de trabalho, é importante ter um **vasto leque de contactos pessoais** fora do grupo e/ou da organização a que pertenço	
G4	Sou capaz de ver como as teorias, os métodos e as técnicas podem ser usadas em **novas relações e situações de trabalho**	
G5	Penso que tenho talento suficiente para **definir os passos concretos** que é preciso dar e estabelecer um bom plano de trabalho	
G6	Estou **atento ao mais pequeno pormenor**, para que nada falhe	
G7	Vejo sempre os dois lados de um problema, de modo a poder tomar uma decisão aceitável por todos	
G8	Em geral, dou-me bem com os outros e esforço-me por **trabalhar com lealdade** em prol da equipa	
Subtotal		20

Este questionário é da autoria do Dr. Luís Graça, pertencendo-lhe todos os direitos sobre o mesmo.

A.2. Interviews' Script

The script for the interviews and the objective behind each question, are detailed below:

1. Tell me a bit about you. – These questions will help the interviewee feel comfortable;
    a. Where are you from?
    b. What experience do you have?
    c. Which technologies favor you?
    d. Where do you see yourself in 3 years? – try to understand his/her commitment to the company;
2. What made you come to the company? – try to understand if he is interested in the job he has, or if it was just a job;
    a. Why did you left your previous employment? – try to understand if he/she has some lessons learned from other companies;
3. How are your relationships inside the organization? – Try to understand if he/she is comfortable at expressing his/her opinion on work related issues;
4. What is your opinion regarding your level of participation in the decision making process? – try to understand if he/she is satisfied with his/her level of participation in the decision making process or if he/she wants more or less;
    a. Would you be interested in changing your level of participation? - this is mainly to access their interest in feeling ownership in their projects;
5. What is your opinion about the schedule commitments for the team and their concretion?
    a. Do you feel they are realistic?
    b. Would you like to be more involved?
6. Do you feel more pressure for delivering work on time or for the quality of your work?
    a. What do you think about the work you produce? – try to understand if he/she is proud of his/hers work;
    b. Do you have any idea of how many defects you produce per KLOC? – try to understand his/her concerns about defects production
    c. Do you review your work?
    d. . How many hours do you think you spend developing code, removing meetings, emails, bathroom and other breaks? – as stated in the book Leadership, Teamwork and Trust: Building a Competitive Software Capability "When they first use the TSP, most teams find that they had been achieving only 10 to 12 task hours a week before they started measuring and tracking their task time;"(Humphrey and Over 2010)
7. How do you feel working here? – see if he/she is happy here

8. How do you feel about your team? – try to understand if he/she trusts his team
9. In your opinion would you change anything? (team, company, work you develop) – try to get some insight on he/she ideas of improvements;
10. Are you excited with this project? – try to understand if he is eager to participate in the pilot project.
    a. Do you have any doubts?
    b. Did you participate in other process changes?
    c. What good or bad experiences did you retained from them?
11. What do you think that makes Alert stand?
12. Any other suggestion?
13. What else should I have asked you?
14. Rate the following topics regarding your satisfaction, from 1 to 10, 1 meaning totally unsatisfied and 10 very satisfied:
    a. Personal work;
    b. Quality of personal work
    c. Work of your team;
    d. Quality of the work of your team;
    e. Team leader;
    f. Development process;
    g. Company;
    h. Company environment;
    i. Global.

Appendix B

Additional Results of the Pilot Implementation

B.1. Pilots Impact

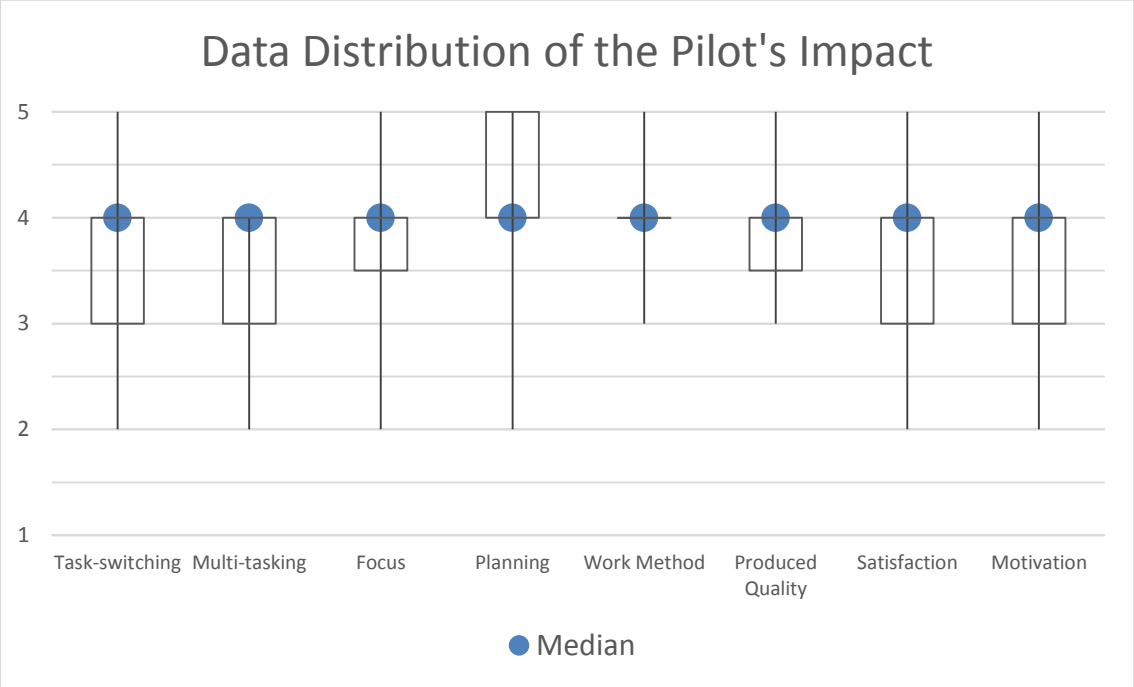

Figure 14 - Data distribution of the pilot's impact - Median, Maximum, Minimum and 1rst and 3rd Quartils

In the charts below – see Figure 15 Figure 16, Figure 17, Figure 18, Figure 19, Figure 20, Figure 21, and Figure 22 – it can be observed the percentage of answers for each of the numbers in the scale of impact, [1, 2, 3, 4, and 5] scale numbers to each of the topics: task-switching; Multi-tasking; Focus; Planning; Work Method; Produced Quality; Satisfaction and Motivation.

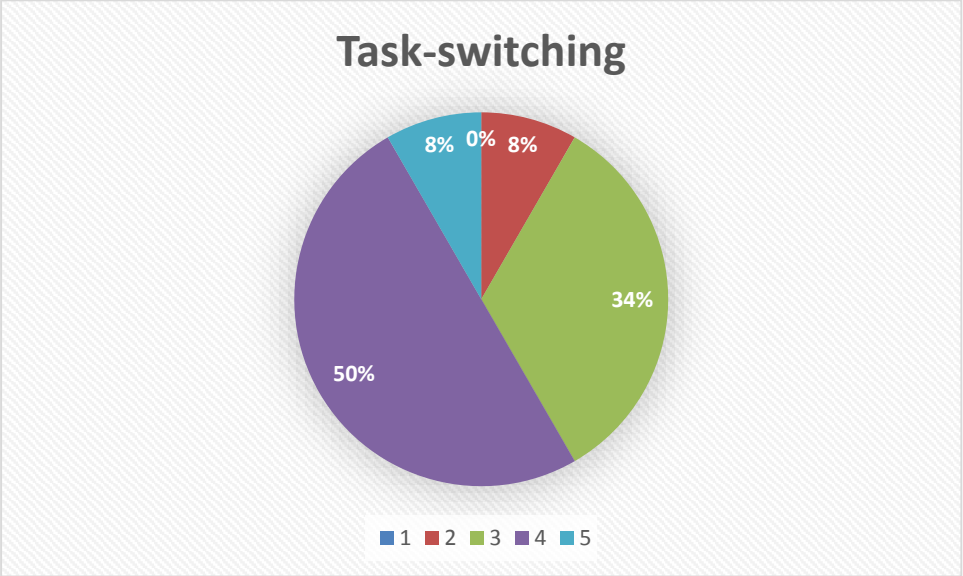

Figure 15 - Percentage of answers to task-switching

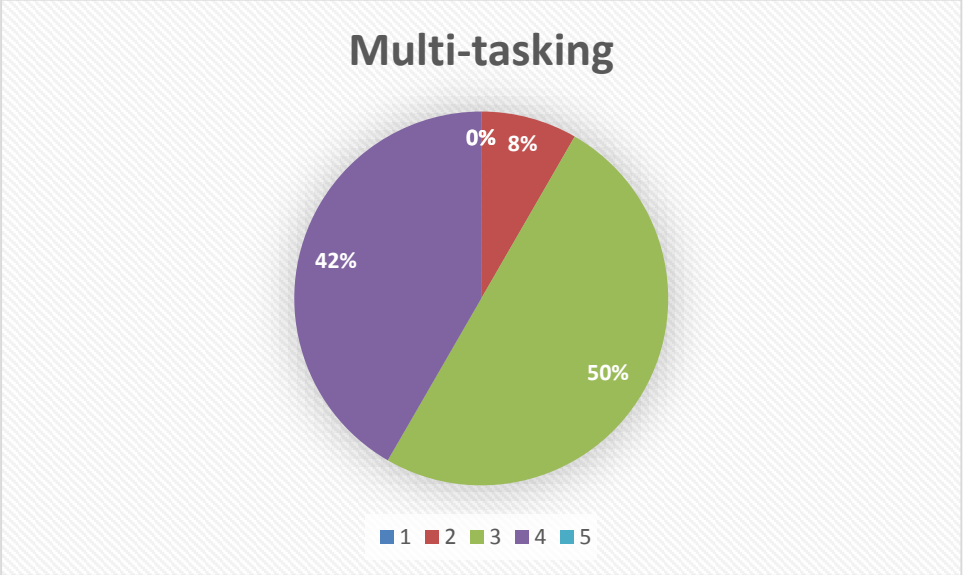

Figure 16 - Percentage of answers to multitasking

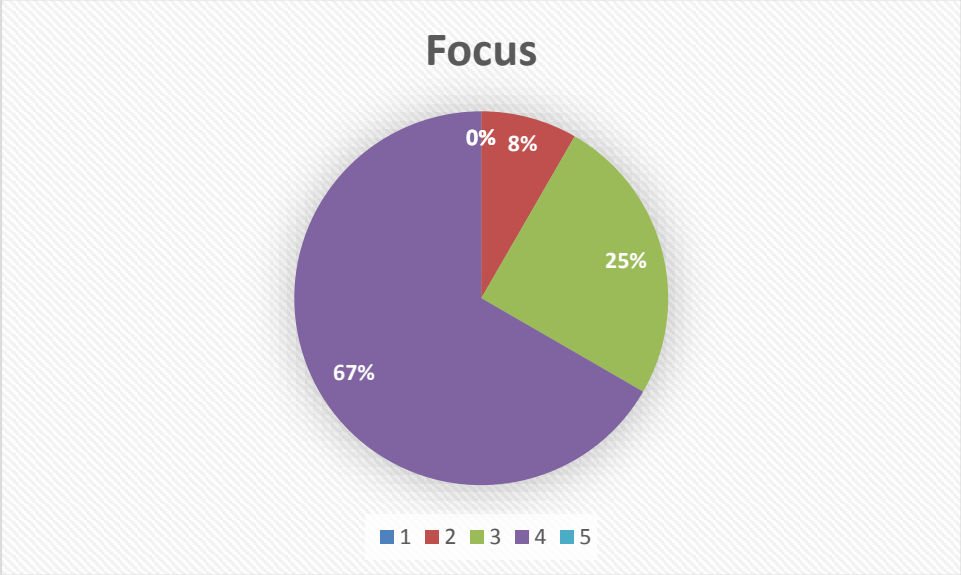

Figure 17 - Percentage of answers to Focus

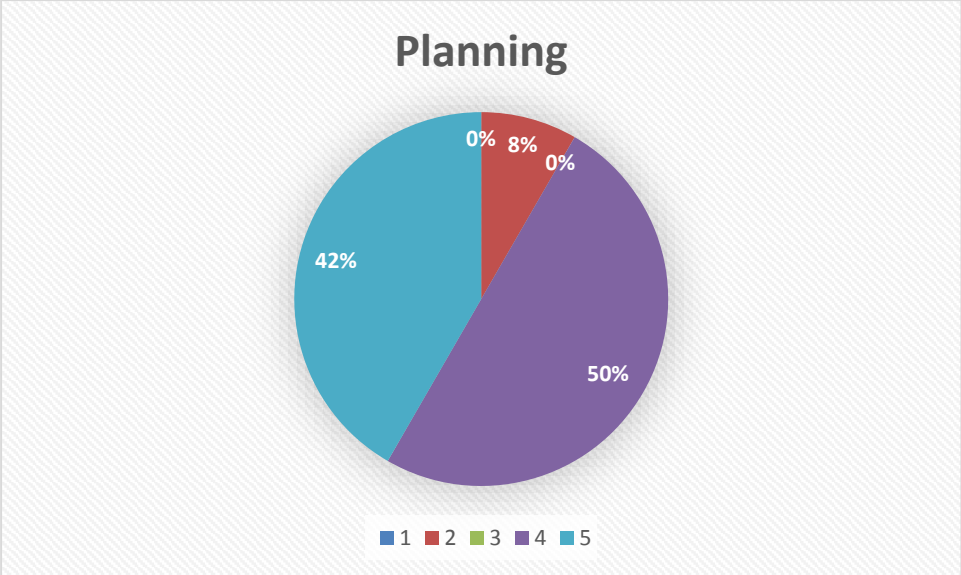

Figure 18 - Percentage of answers to Planning

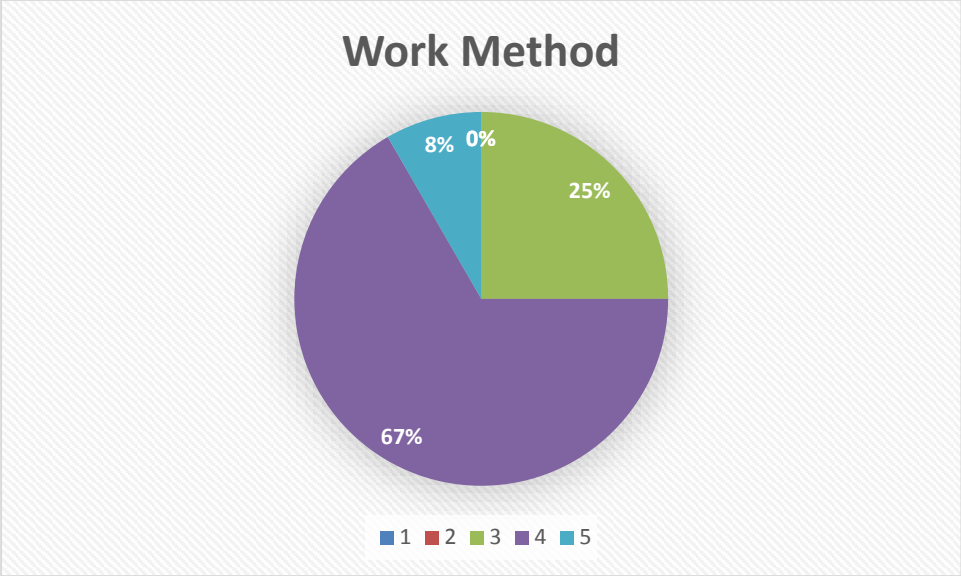

Figure 19 - Percentage of answers to Work method

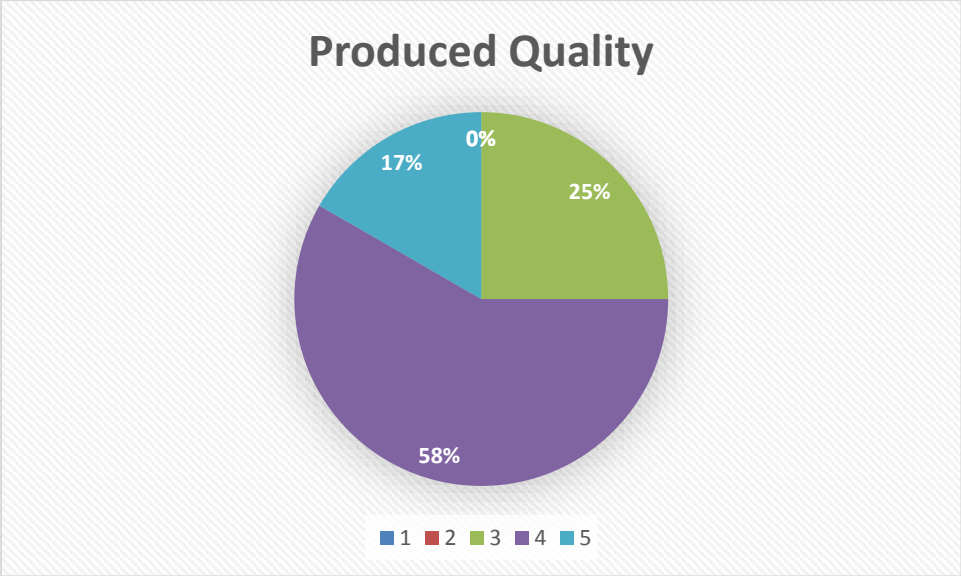

Figure 20 - Percentage of answers to produced quality

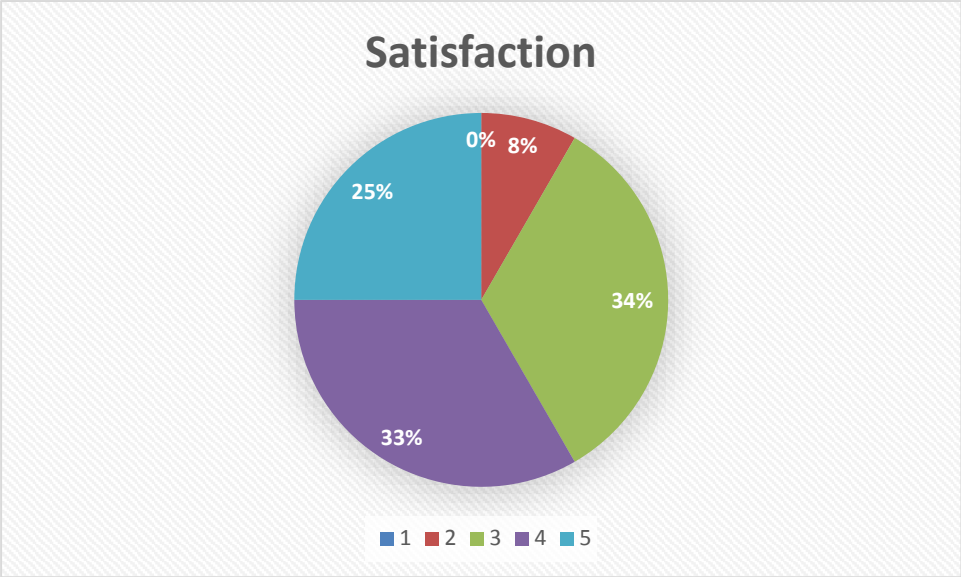

Figure 21 - Percentage of answers to satisfaction

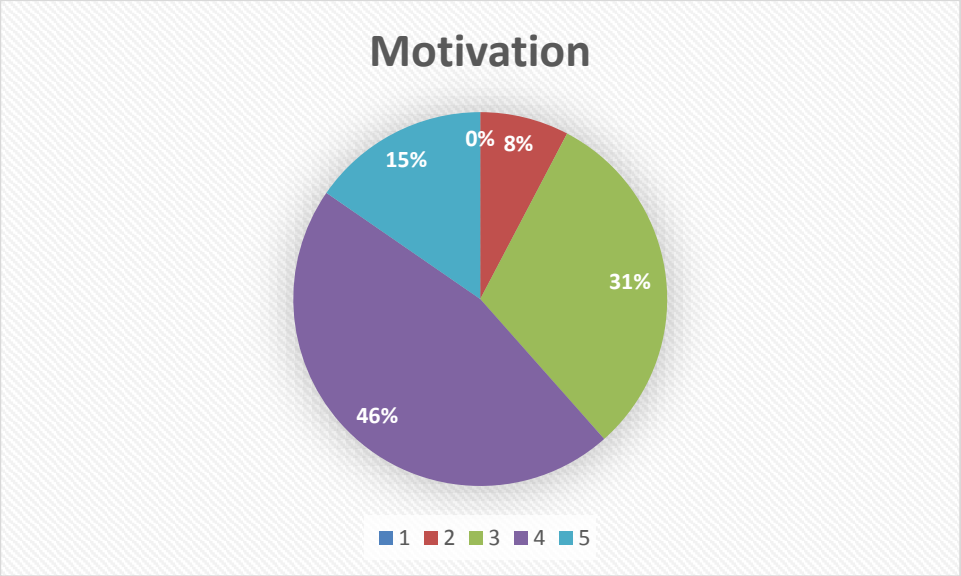

Figure 22 - Percentage of answers to Motivation

B.2. Estimates

For each Team in the first chart we will be able to see the error for each estimate in each Sprint. Being that the percentage will correspond to the error in estimate and the number correspond to the difference in man-days. In Team per example we will see that for the first item in Sprint 1 (first A1) the Team only spent near 20% of the estimate, which represented that they used 2.5 man-days less than what was estimated.

In the second chart, for each team, we will see the data distribution of estimates for each Sprint. Representing Median, Minimum, Maximum, and first and third Quartiles.

To identify the Sprint it was used the letter that identifies the team and the number that identifies the Sprint.

B.2.1. Team A

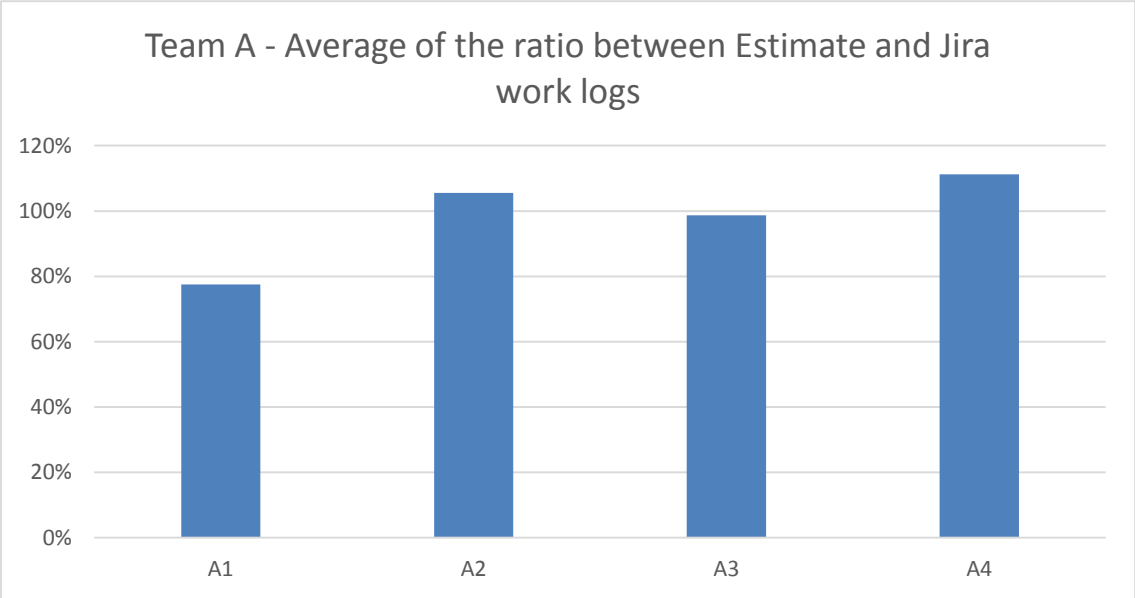

Figure 23 - Average of the ratio of Team A's worklogs and estimates in each Sprint

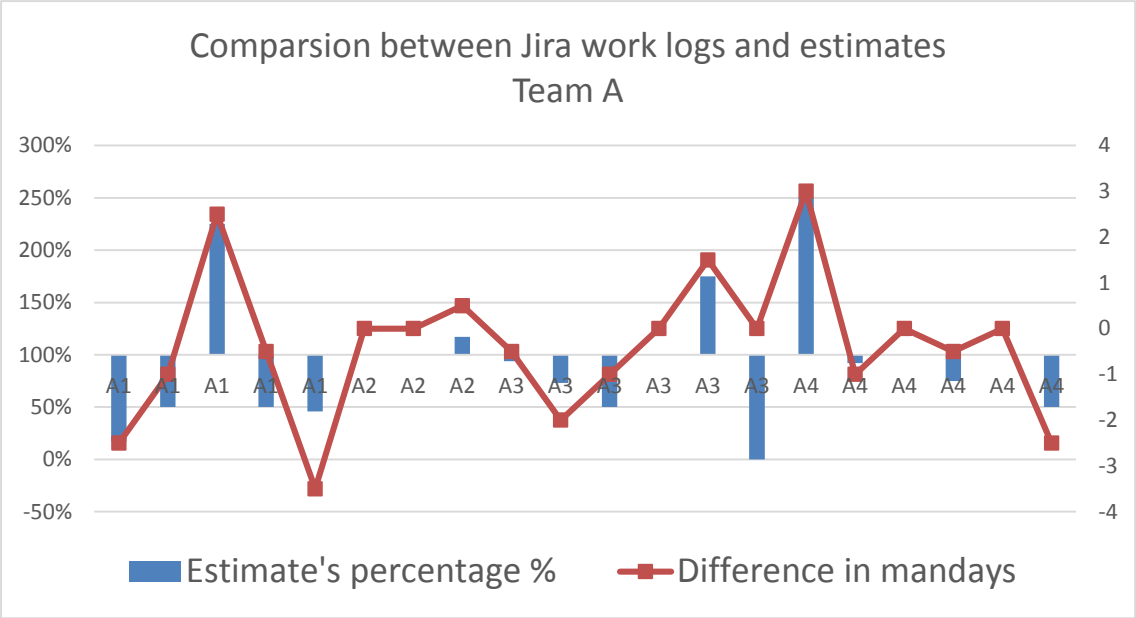

Figure 25 - Team A - Differents between estimates and man-days for each item in each Sprint

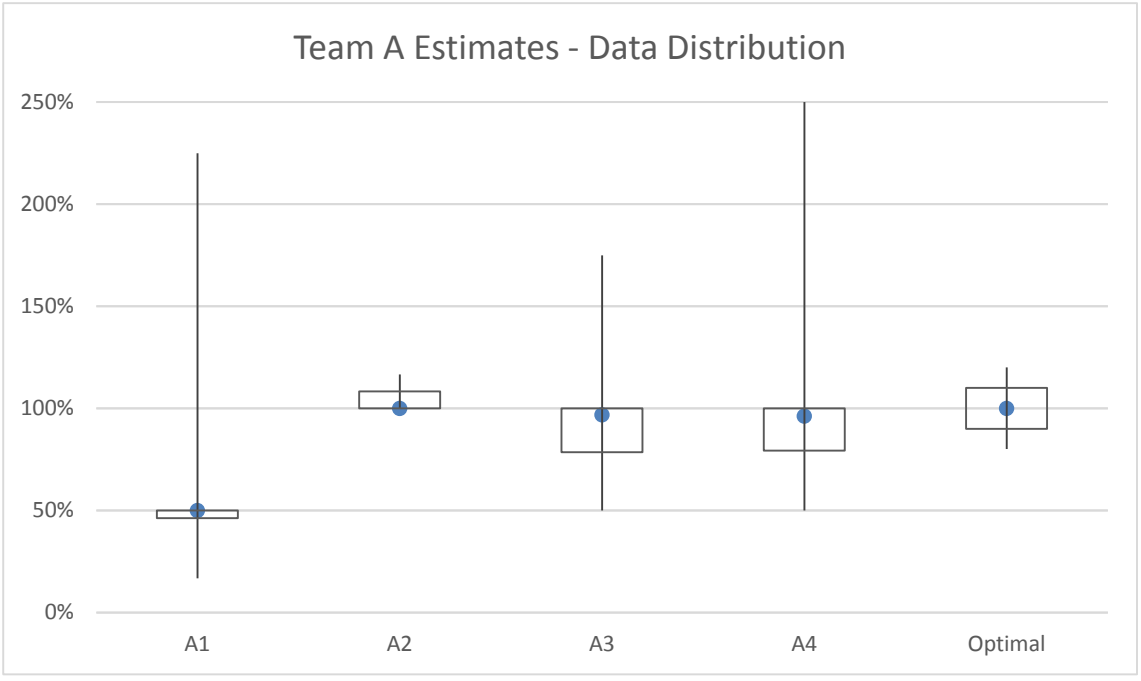

Figure 24 - Data distribution of Team A's estimates for the 4 Sprints, and representing the considered optimal

B.2.2. Team B

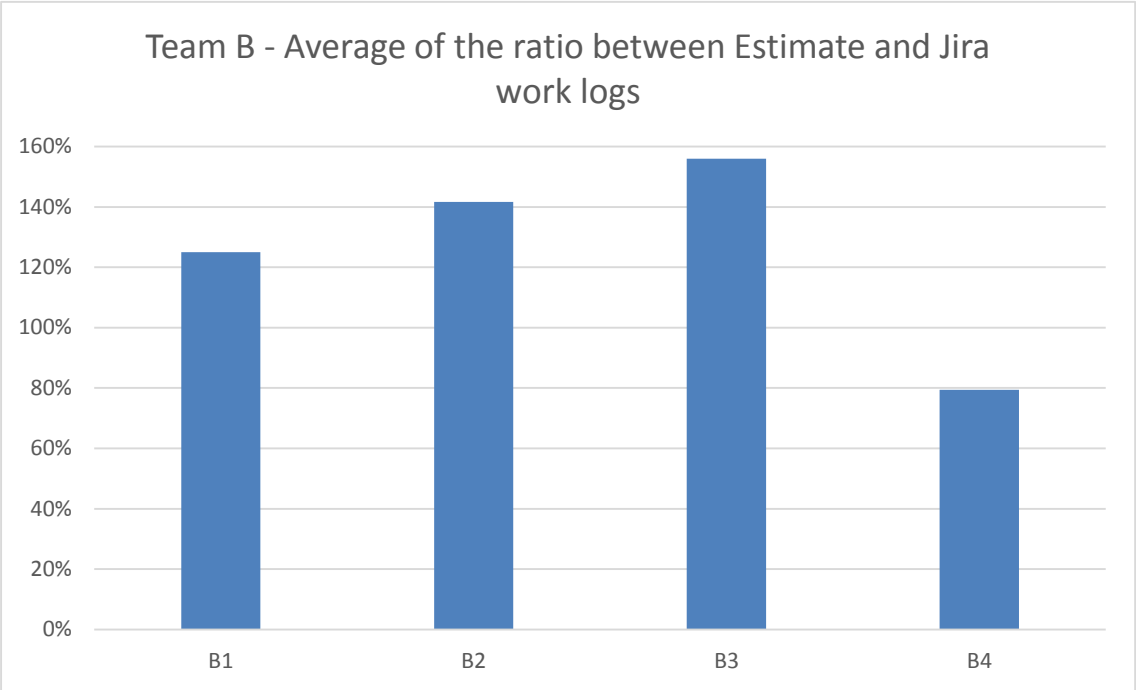

Figure 27 - Average of the ratio of Team B's worklogs and estimates in each Sprint

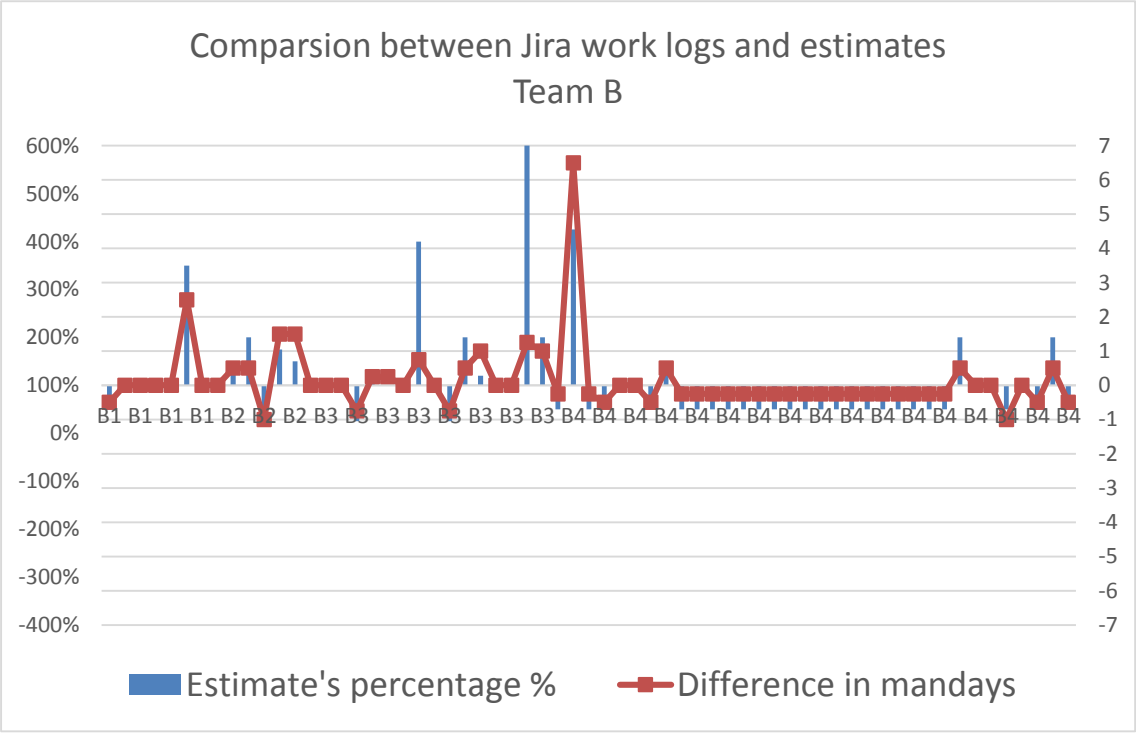

Figure 26 - Team B - Differences between estimates and man-days for each item in each Sprint

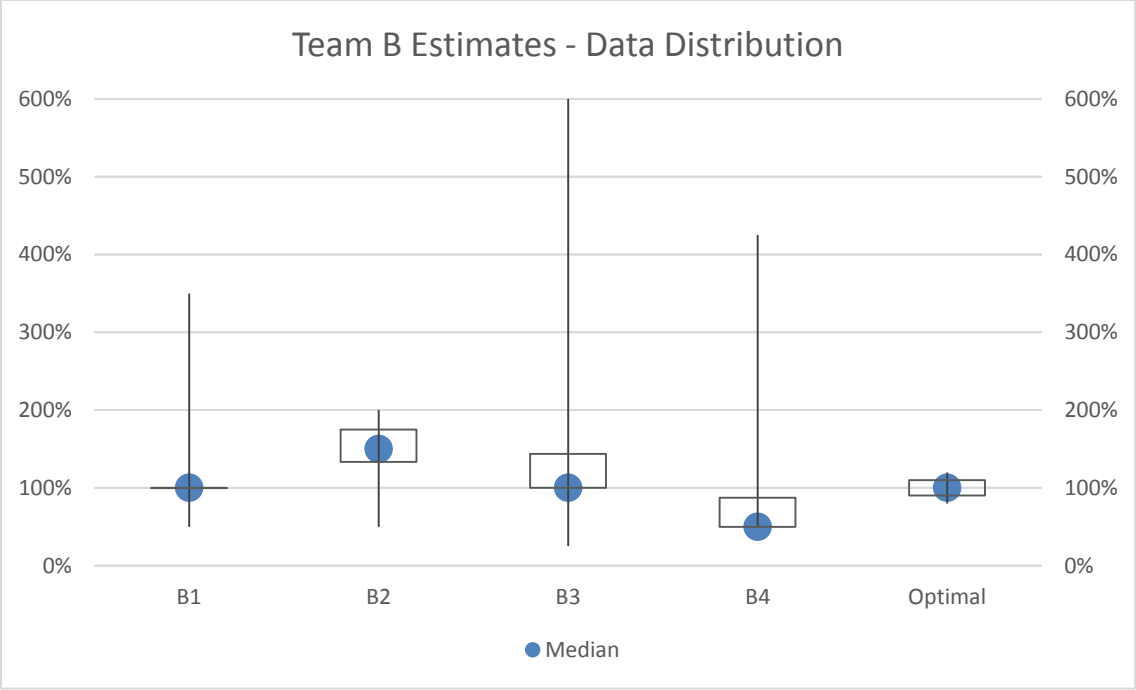

Figure 29 - Data distribution of Team B's estimates for the 4 Sprints, and representing the considered optimal

B.2.3. Team C

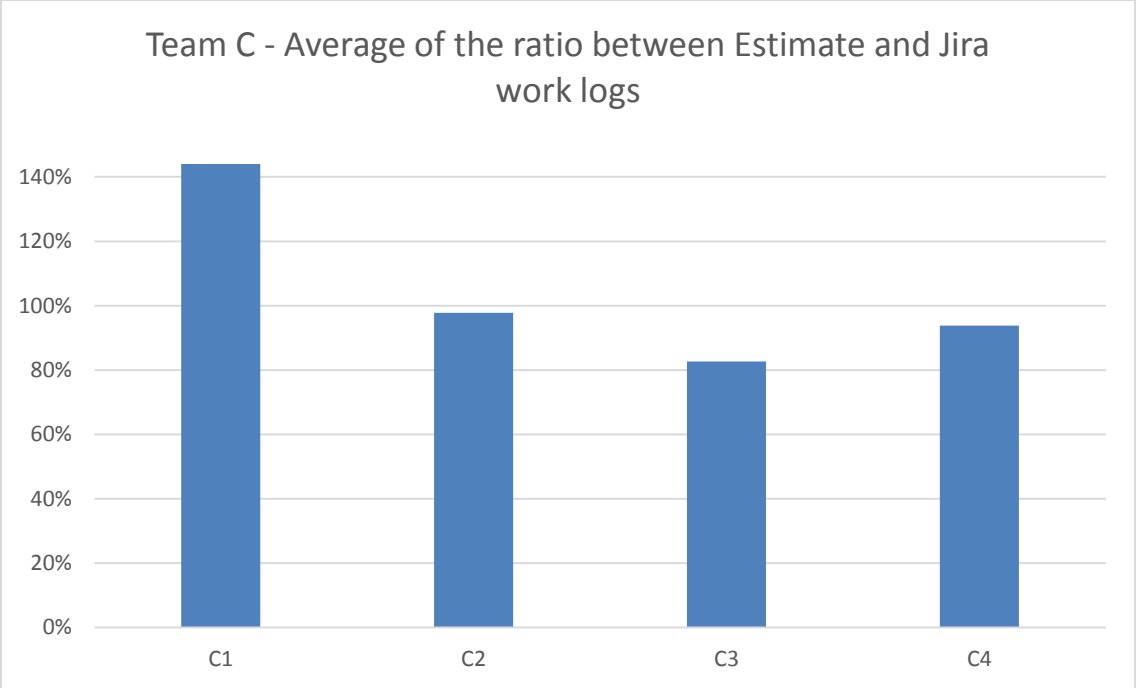

Figure 28 - Average of the ratio of Team C's worklogs and estimates in each Sprint

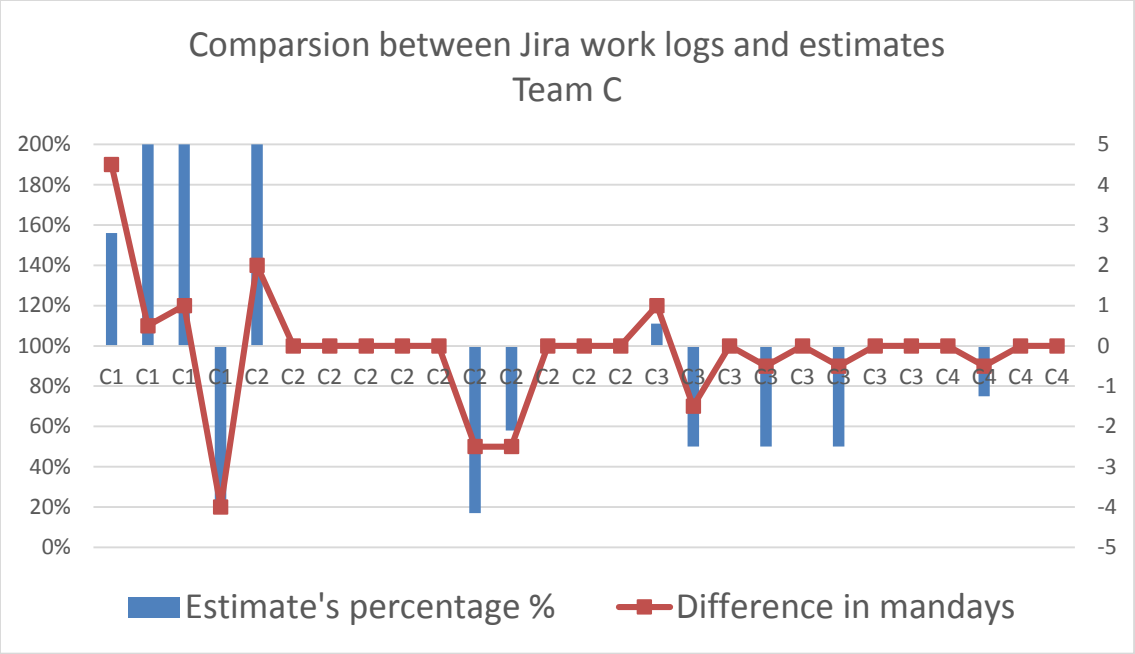

Figure 31 - Differences between estimates and man-days for each item in each Sprint

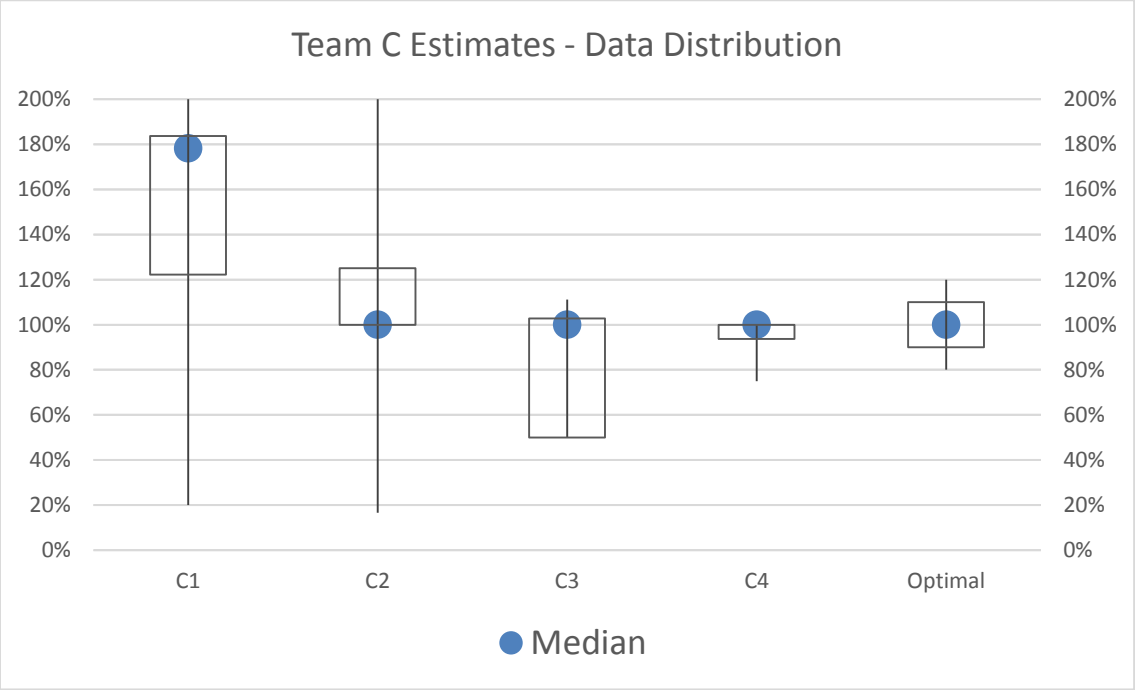

Figure 30 - Data distribution of Team C's estimates for the 4 Sprints, and$_{nd}$ representing the considered optimal